目　录

第 1 章　MATLAB 6 简介

MATLAB 6 是一套功能十分强大的工程计算及数据分析软件，它的应用范围覆盖了工业、电子、医疗、建筑等众多领域。今天的工程师们都面临着如何在短时期内高效出色地完成复杂的科研项目的难题。MATLAB 6 是一种交互式、面向对象的程序设计语言，其结构完整，具有优良的移植性。它主要用于矩阵运算，同时在数据分析、自动控制、数字信号处理、绘图等方面也具有强大的功能。许多工程师发现，它能迅速地验证他们的构思，综合评测系统性能，并能快速设计出更多解决方案来确保未来更高的技术要求。

1.1　程序设计环境

本节对 MATLAB 6 的程序设计环境进行了详细地介绍，主要包括工作环境、命令窗口的设置、M 文件的编辑调试环境与 MATLAB 的搜索路径。

1.1.1　MATLAB 的工作环境

MATLAB 的工作环境简单、明了，易于操作。

1.　命令窗口

启动 MATLAB 后，显示的操作界面如图 1.1 所示。



图1.1　MATLAB的操作界面

命令窗口是 MATLAB 的主窗口，用户可以通过单击命令窗口右上角的按钮使其放大成一个独立的窗口。在命令窗口中可以直接输入命令，系统将自动显示信息，例如，在命令窗口中输入指令：

```
x=[2 3 5;4 3 2 ;7 8 9;6 3 2;2 3 4;3 4 5;6 8 9]
```

数据放在方括号内，行与行之间用分号间隔，数值之间用空格或逗号间隔。如果命令后不加“;”，则系统解释该命令为一个 7×3 的矩阵，并显示如下结果：

```
x=
    2     3     5
    4     3     2
    7     8     9
    6     3     2
    2     3     4
    3     4     5
    6     8     9
```

若程序有多行语句，且不需要每行都显示结果，可在不需显示结果的语句后加上“;”，这在编写 M 文件时非常有用。

如果一条语句过长，需要两行或多行才能输入，则要使用“…”作连接符号，按 Enter 键转入下一行继续输入。另外，在命令窗口输入命令时，可利用快捷键或功能键方便地调用或修改以前输入的命令。如通过“↑”键可重复调用上一个命令行，对它加以修改后重新执行，而且在执行命令时，不需将光标移至行尾。表 1.1 列出了 MATLAB 中常用的命令行快捷键与功能键。

表 1.1 MATLAB 中常用的命令行快捷键与功能键

功能键	快捷键	功能说明
↑	Ctrl+P	重新调出上一行
↓	Ctrl+N	返回下一行输入
←	Ctrl+B	光标左移一个字符
→	Ctrl+F	光标右移一个字符
Ctrl+→	Ctrl+R	光标右移一个字
Ctrl+←	Ctrl+L	光标左移一个字
Home	Ctrl+A	光标移至行首
End	Ctrl+E	光标移至行尾
Esc	Ctrl+U	清除命令行
Del	Ctrl+D	删除光标处字符
Backspace	Ctrl+H	删除光标前一处字符
—	Ctrl+K	删除至行尾
—	Ctrl+C	中断正在执行的命令

可以用 clc 命令清除命令窗口显示的内容，但是需要注意的是，此命令并不清除工作空间，而仅清除窗口显示，通过↑键仍可返回前一行输入的指令。

可以用 format 命令来控制命令窗口中数值显示的格式，或者选择 File | Preferences 命

令，打开 Preferences 对话框，选择如图 1.2 所示的 Command Window 目录，并在 Numeric format 下拉列表框中选择所需的数值显示格式。表 1.2 列出了 MATLAB 中不同的数值显示格式及其范例。

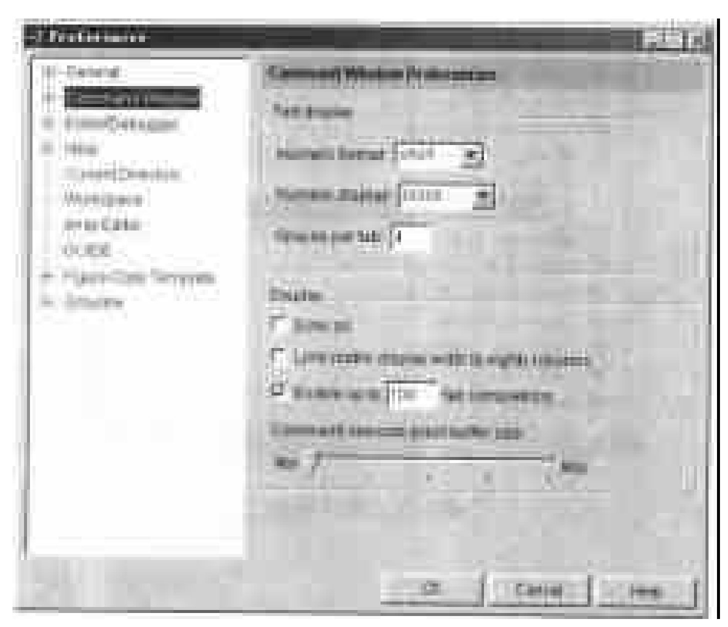


图1.2 控制命令窗口中数值显示格式

表 1.2 数值显示格式及其范例

显示格式	范例 1 (4/3)	范例 2 (2.45563652e-4)
Short	1.3333	2.4556
Short E	1.3333e+000	2.4556e-004
Short G	1.3333	2.4556e-004
Long	1.33333333333333	2.455636520000000e-004
Long E	1.33333333333333 e+000	2.455636520000000e-004
Long G	1.33333333333333	0.000245563652
Bank	1.33	0.00
Rat	4/3	19/77373
Hex	3ff5555555555555	3f3017dfdabbaf03

上述显示格式都可以在命令窗口中直接实现，如输入 format long。另外，命令 format loose 与 format compact 用于控制显示的数值之间是否换行。

2. 启动平台

当用户需要启动某个工具箱的应用程序时，可以在 Launch Pad(启动平台)中实现，当单击 Launch Pad 窗口的 按钮后，启动平台就最大化，如图 1.3 所示。此时用户可以方便地打开工具箱中的内容，包括帮助文件、演示示例、实用工具以及 Web 文档。例如要启动 Filter Design Toolbox(滤波器设计工具箱)的实用工具界面 Filter Design & Analysis Tool(FDATool)，可以双击该项目。



图1.3 启动平台

3. 工作空间

工作空间是 MATLAB 6 的新特点，以前的工作空间只是一个对话框，可操作性差，现在的工作空间作为一个独立的窗口，其操作相当方便。当单击工作空间窗口的按钮后，工作空间就最大化，如图 1.4 所示。



图1.4 工作空间窗口

4. 命令历史记录与当前路径窗口

命令历史(Command History)记录窗口主要显示已执行过的命令，利用表 1.1 所讲的功能键或快捷键，可以非常方便地重复调用命令。当前路径窗口主要显示当前工作在什么路径下，包括 M 文件的打开路径等。

1.1.2　命令窗口的设置

当使用命令窗口进行工作时，用户可以根据自己的习惯与要求，来设置命令窗口的显示方式。

设置命令窗口时，首先要选择 File | Preferences 命令打开如图 1.5 所示的参数设置对话框，单击 Command Window 标签即可进入命令窗口的设置。

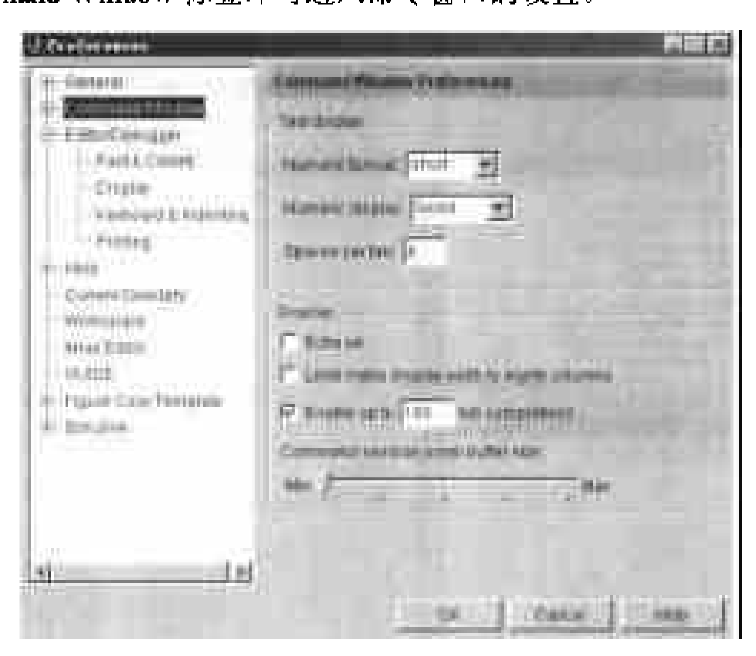


图1.5　命令窗口设置对话框

- Text display 选项组

  该选项组用来设置命令窗口中的数据格式、窗口数字显示与 Tab 制表符的字符数，数据格式设置 Numeric format 下拉列表框的使用可参考 1.1.1 节的有关内容。

  - Numeric display 下拉列表框

    用来设置命令窗口的数值显示，选择 Compact 选项表示以文字紧缩形式显示，选择 Loose 选项表示以文字宽松形式显示。

  - Spaces per tab 文本框

    用来设置 Tab 制表符的宽度。

- Display 选项组

  - Echo on 复选框

    在执行 M 文件时，如果想将执行的命令显示在命令窗口，则可以选中该复选框。

  - Limit matrix display width to eighty columns 复选框

    如果想在命令窗口中显示 80 列输出，而不管命令窗口的实际宽度为多大，则可以选中该复选框。如果不选择该项，则可以使命令窗口更宽，并且可以使输出填满命令窗口宽度。

  - Enable up to [n] tab completions 复选框

如果选中该复选框，则可在命令窗口输入函数时使用 Tab 键完成功能。

◆ Command session scroll buffer size 滑杆
  用来设置命令窗口中卷轴缓冲器的大小。

1.1.3 M 文件的编辑调试环境

MATLAB 的 M 文件通常保存为后缀为“.m”的文件，MATLAB 具有自身的 M 文件编辑器与调试器(Editor | Debugger)，如图 1.6 所示。



图1.6 M文件的编辑与调试环境

1. Editor/Debugger的参数设置

当使用 MATLAB 编辑器与调试器进行工作时，常常需要设置一些适合自己需要的环境，此时可以选择 File | Preferences 命令来进行。执行命令后，系统将弹出如图 1.7 所示的 Preferences 对话框。在该对话框中，可以设置各个窗口的参数，包括命令窗口、Editor/Debugger 窗口、工作空间等。

在 Preferences 对话框中，单击 Editor/Debugger 目录后就进入如图 1.7 所示的环境，在此即可设置 Editor/Debugger 的参数。如果双击该目录则弹出 Editor/Debugger 的参数设置子标签，用户还可以设置如下有关 Editor/Debugger 的基本参数：

- Editor 选项组
  该选项组的选项用来设置用户将要使用的文本编辑器。
  ◆ 选中 Built-in editor 单选按钮表示使用 MATLAB 的内置编辑器。
  ◆ 选中 Other 单选按钮表示可以选择其他编辑器，此时可以在文本框中输入编辑器的路径及应用程序名称。
- Debugger Options 选项组
  该选项组用来设置是否允许在命令窗口进行调试，选中 Command Window

debugging 复选框则表示可在命令窗口执行调试功能。

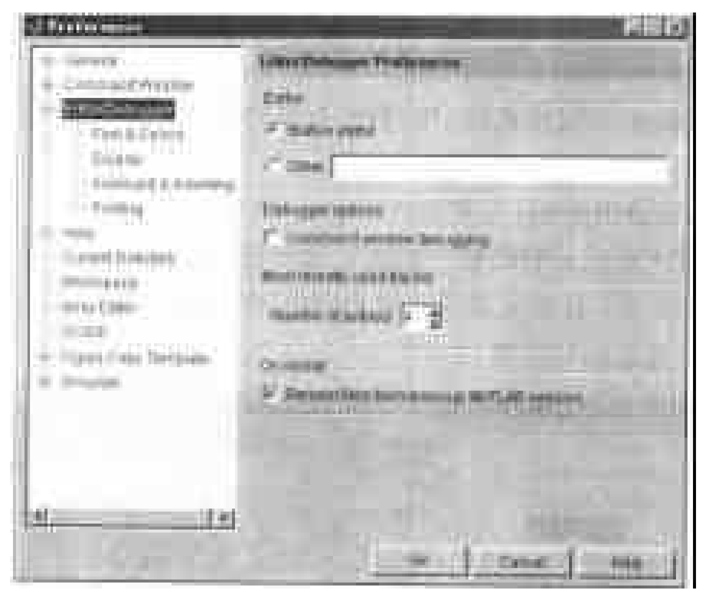


图1.7 参数设置对话框

- Most recently used files list 选项组

  该选项组与其他文本编辑器类似，用来设置最近使用的文件列表数目，其最大设置数目为 9，使用户可以方便地加入最近使用的文件。

- On restart 选项组

  该选项组用来设置当重新运行系统时，是否打开原来操作的文件。若选中 Reopen files from previous MATLAB session 复选框，则表示下次启动 MATLAB 时，打开上一次退出 MATLAB 时正在编辑调试的文件。

2. 设置字体与颜色

用户可以方便地在 Editor/Debugger 的参数设置对话框中设置字体与颜色，只需单击对话框中的 Font & Colors 标签，系统就会弹出如图 1.8 所示的字体与颜色设置对话框。

- 字体设置

  Font 选项组用来设置字体。

  - Use desktop font 单选按钮

    若选中该单选按钮，则表示 Editor/Debugger 窗口中的字体采用 Windows 桌面字体，且选中该项后，Font 操作框中的其他选项成为灰色，不能再被选中。

  - Use custom font 单选按钮

    用户可以使用该单选按钮来设置自己喜欢的字体，包括字体的类型与大小，比如可以设置 Editor/Debugger 窗口中的字体为宋体 12 号等。

- 颜色设置

  Colors 选项组用来设置颜色。

  - Text color 下拉列表框

    如果在 Text color 下拉列表框中将颜色设置为 Automatic，系统会自动使用

默认的颜色设置，且不可设置背景颜色。若采用其他的颜色设置，则可以设置背景颜色。

- ◆ Backgroud color 下拉列表框

  Text color 下拉列表框中的颜色不设置为 Automatic 时，用户可以调整背景颜色。

- ◆ Syntax hightlighting 复选框

  如果想区别显示编辑框中的语法项与其他语句，则可选中该复选框，使语法项高亮显示。若想设置高亮显示颜色，可以单击 Set Colors 按钮，进入 General 选项卡设置此项功能。

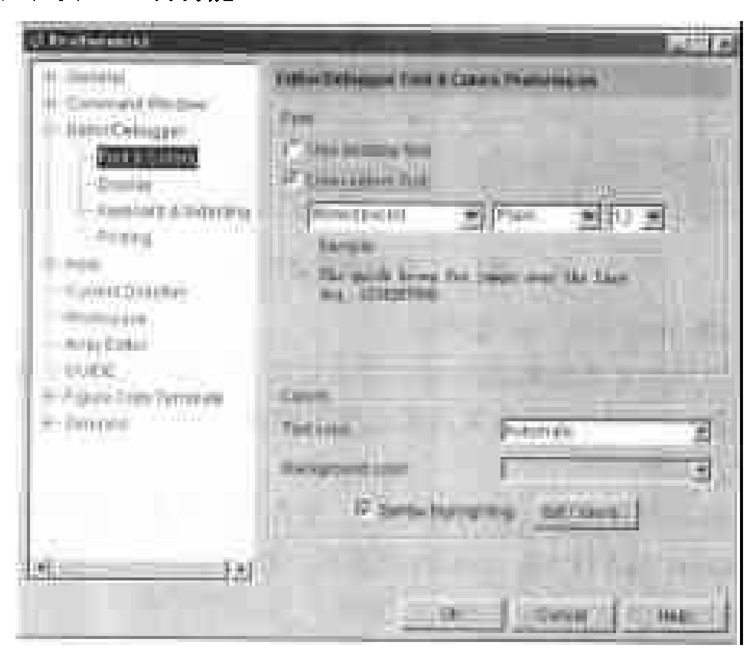


图1.8 "字体与颜色"设置对话框

3. 设置显示方式

单击如图 1.9 所示的 Display 标签，系统自动弹出如图 1.9 所示的对话框，可以分别设置文件的打开方式和 Editor/Debugger 中的工具显示。

- ● 设置文件的打开方式

  可以通过 Opening files in eitor 选项组来设置文件的打开方式，此选项组共有两个互锁选项。

  - ◆ Single window contains all files(tabbed style)单选按钮

    该选项表示在一个窗口中显示多个文件，各个文件以标签的形式显示在左下角。

  - ◆ Each file is displayed in its own window 单选按钮

    该选项表示每个文件在各自独立的窗口中显示。

- ● 设置 Editor/Debugger 中的显示

  可以通过 Display 选项组来设置 Editor/Debugger 中的显示，此选项组共有 3 个选项。

- ◆ Show toolbar 复选框
  在 Editor/Debugger 中显示工具栏。
- ◆ Show line numbers 复选框
  选中此复选框后可在 Editor/Debugger 中显示文本的行数，这在修改与调试 M 文件时非常有用。
- ◆ Enable datatips in edit mode 复选框
  选中此复选框后可显示数据提示，即在编辑窗口中，当用户用鼠标指针指向某个变量时，系统会自动显示该变量的内容。

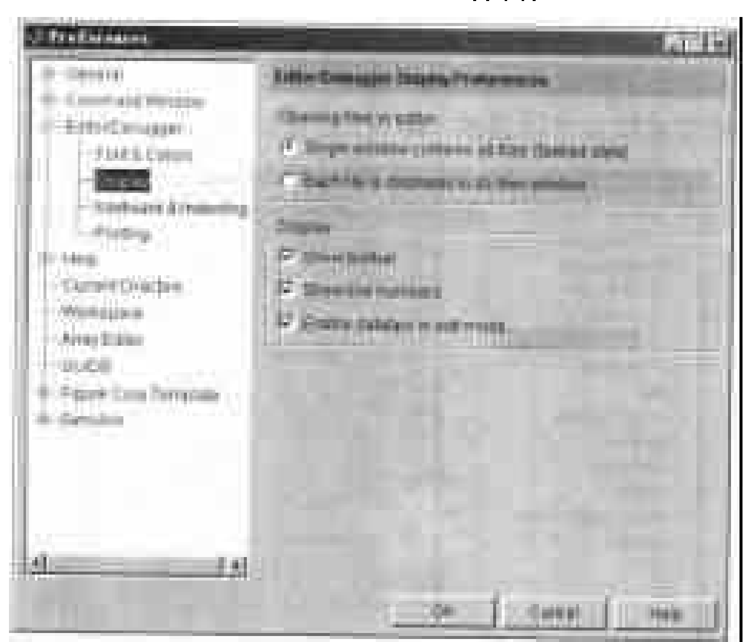


图1.9 Editor/Debugger显示设置对话框

4. 设置键盘与缩进

单击如图 1.10 所示的 Keyboard & Indenting 标签，系统自动弹出如图 1.10 所示的键盘与缩进设置对话框，可以分别设置键盘操作和字符缩进。

- ● 键盘设置
  用户可以通过 Key bindings 选项组设置适合自己习惯的键盘定义，此选项组共有两个互锁选项。
  - ◆ Windows 单选按钮
    使用 Windows 系统约定的键盘快捷定义，如复制的快捷键为 Ctrl+C，粘贴的快捷键为 Ctrl+V。
  - ◆ Emacs 单选按钮
    使用 Emacs 约定的键盘快捷定义，如粘贴的快捷键为 Ctrl+Y。
- ● M 文件缩进设置
  用户可以通过 M-file indenting for Entey key 选项组设置 M 文件的不同缩进格式，此选项组共有 3 个互锁选项。
  - ◆ No indent 单选按钮

文本无缩进格式。

- ◆ Block indent 单选按钮
  块形式缩进格式。
- ◆ Smart indent 单选按钮
  智能缩进格式。

- ● 缩进参数设置

用户可以通过 Indents 选项组设置适合自己的缩进参数。

- ◆ Indent size 文本框
  可输入同一标准的嵌套代码列数。
- ◆ Emacs style Tab key smart indengting 复选框
  选中此复选框后可以通过 Tab 键缩进当前行。

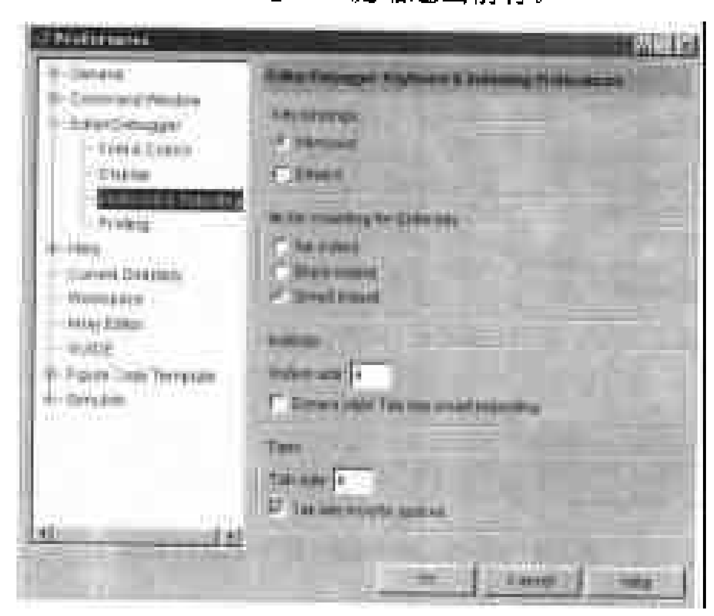


图1.10 键盘与缩进设置对话框

- ● 制表符设置

用户可以通过 Tabs 选项组设置适合自己的制表符参数。

- ◆ Tab size 文本框
  设置两个制表符 Tab 间的空格数。与 Auto indent size 中设置的数目不同，自动缩进只是插入空格，不插入制表符 Tab。
- ◆ Tab key inserts spaces 复选框
  选中此复选框后可选择插入一个 Tab 字符，或相当于一个 Tab 字符的空格数目。

1.1.4 MATLAB 的搜索路径

MATLAB 采用路径搜索的方法来查找文件系统中的 M 文件，常用的命令文件组在 MATLAB 文件夹中，其他 M 文件组在各种工具箱中。如果在命令窗口中输入如下命令：

```
test
```

MATLAB 对这一命令的搜索顺序为：

(1) 检查 test 是否为存储在工作空间中的变量，若为工作空间中的变量，则返回该变量的内容。

(2) 检查 test 是否为 MATLAB 的内部函数，若为内部函数，则返回要求输入内部函数参数的信息。例如在命令窗口中输入"fft"，则得到下面的反馈信息：

```
???Error using ==> fft
Not enough input arguments.
```

(3) 检查当前目录中是否有 test.m、test.mex 或 test.dll 文件。

(4) 检查 MATLAB 搜索路径上是否存在 test.m、test.mex 或 test.dll 文件。

(5) 如不满足上述任何一条件，则返回出错信息。

如果在搜索路径中存在两个或多个同名函数，则只能发现搜索路径中的第一个函数，而其他同名函数不被执行。此搜索的顺序只是一般情况下的顺序，而实际的搜索规则要复杂得多，例如，要考虑是否有私有函数、子函数以及面向对象的函数等限制。

使用下面的一组命令可以对当前的搜索路径进行操作：

- 在命令窗口中只输入命令"path"，而不加任何参数，则得到当前的搜索路径。
- 输入"path+路径名"，设置当前的搜索路径，以前的搜索路径无效。
- 利用下述命令可以向当前搜索路径中添加路径：

```
addpath c:\MATLAB\work
addpath 'Macintosh HD:MATLAB:My Tools'
addpath /home/user/MATLAB
```

- 命令 rmpath 可以取消当前搜索路径中的目录：

```
rmpath c:\MATLAB\work
rmpath 'Macintosh HD:MATLAB:My Tools'
rmpath /home/user/MATLAB
```

- 命令 pathtool 调用如图 1.11 所示的 Set Path(设置路径)对话框。

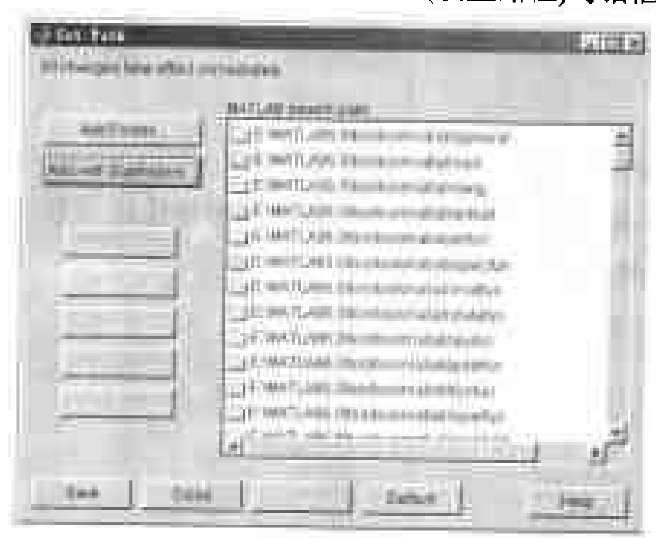


图1.11　Set Path对话框

在 Set Path 对话框中，可以使用 Move Up、Move Down、Move to Top、Move to Bottom

等按钮调整搜索路径的顺序。使用 Remove 按钮可以删除选中的搜索路径。

单击 Add Folder 按钮则打开如图 1.12 所示的【浏览文件夹】对话框，选择要添加的目录。在 Set Path 对话框中还可以单击 Add with Subfolders 按钮，将选中的目录路径的子目录也包含在搜索路径中。



图1.12 【浏览文件夹】对话框

1.2 基本操作

通过上节的学习，我们对 MATLAB6 的程序设计环境有了清楚的了解。在此基础上，本节对其基本操作进行阐述，主要包括一些常用的基本知识、矩阵运算及分解、数据分析与统计。

1.2.1 基本知识

1. 矩阵的输入

MATLAB 的基本操作对象是矩阵。在 MATLAB 下输入矩阵有 4 种方式：

(1) 在 MATLAB 命令窗口中直接输入，这是最方便快捷的方式。

例如，在 MATLAB 的命令窗口中输入：

```
x=[1 4 5; 2 4 6;3 5 8]
```

则其输出结果为：

```
x =
   1     4     5
   2     4     6
   3     5     8
```

(2) 利用内部语句或函数产生矩阵。

例如，在 MATLAB 的命令窗口中输入：

```
x=randn(4,5)
```

则其输出结果为：

```
x =
        -0.4326    -1.1465     0.3273    -0.5883     1.0668
        -1.6656     1.1909     0.1746     2.1832     0.0593
         0.1253     1.1892    -0.1867    -0.1364    -0.0956
         0.2877    -0.0376     0.7258     0.1139    -0.8323
```

利用内部语句和函数可以快速产生矩阵，并且可以得到一些特别有用的矩阵，如单位阵、稀疏矩阵等。

(3) 利用 M 文件产生矩阵。

对于大的且经常调用的矩阵，可以将其存入 M 文件中，以方便调用，避免重复输入。如何编写 M 文件的内容可以参考 1.4 节。

(4) 利用外部数据文件装入到指定矩阵。

利用 load 命令或 fscanf 命令可以读取早期版本所产生的矩阵，也可读取由其他应用程序产生的数据。

例如，有一数据文件的存放目录为 d:\seaclutter\seadata\data.dat，输入下面的语句便可将数据调出，并自动存放在 data 变量中。

方法 1：利用 load 命令。

```
load d:\seaclutter\seadata\data.dat
data
```

方法 2：利用 fscanf 命令。

```
fid=fopen('d:\seaclutter\seadata\data.dat','r');
data=fscanf(fid, '%f' ,[4,5])
status=fclose(fid);
```

输出结果为：

```
data =
          0.2944     1.1908     0.7119    -0.3999    -1.5937
         -0.6918    -1.3362    -1.2025     1.2902     0.6900
         -1.4410     0.8580     0.7143    -0.0198     0.6686
          0.8156     0.5711     1.2540     1.6236    -0.1567
```

其中在方法 2 中，如果文件不能打开，则 fopen 返回−1，这时可以借助第 2 个参数来获得有关出错信息，格式为：

```
[fid,message]=fopen('d:\seaclutter\seadata\data.dat','r')
```

参数 message 中返回有关出错的提示信息。

2. 复数与复数矩阵

MATLAB 中利用特殊的字符 i 或 j 来表示复数，也就是说，字符 i 或 j 相当于复数中的虚数单位，且字符 i 与 j 的效果是一样的。

输入矩阵有两种方法：

(1) x=[3 5 7;2 4 6]+i*[1 4 3;4 5 6]

(2) x=[3+i 5+4i 7+3i;2+4i 4+5i 6+6i]

这里需要注意的是，如果在一个 M 文件中字符 i 或 j 被赋了值，这时 i 或 j 被当作变量使用，不再是虚数单位。如果需要虚数单位，可输入下列语句：

```
i1=sqrt(-1)
```

3. 固定变量

MATLAB 中提供了一些用户不能清除的固定变量，如 ans、pi、Inf、NaN。

- ans：在没有定义变量名时，系统默认变量名为 ans。
- pi：变量 pi 即为圆周率 π。
- Inf：表示无穷大。当作除法运算时，若分母为零，就会出现如下警告信息：

```
Warning: Divide by Zero
ans=
Inf
```

- NaN：变量 NaN 表示不定值，它由 Inf/Inf 或 0/0 运算产生。

4. 获取工作空间信息

MATLAB 中有一专门的工作空间，用于存放已经产生的变量。要获取工作空间的信息，可以通过下述命令来实现。

- who 命令

该命令用于显示工作空间中保存的变量名。如在命令窗口中输入：

```
who
```

则显示工作空间中保存的变量名：

```
Your variables are:
a          c          y
b          x          z
```

- whos 命令

该命令用于显示工作空间中各变量的属性，包括大小、元素个数、所占用的字节数、元素精度。如在命令窗口中输入：

```
whos
```

则产生：

```
Name    Size        Bytes   Class
   a    1x32        256     double array
   b    12x32       3072    double array
   c    6x32        1536    double array
   x    16x6        768     double array
   y    16x10       1280    double array
   z    6x10        480     double array
Grand total is 924 elements using 7392 bytes
```

5. 函数

MATLAB 具有强大的函数功能，从本质上讲，可将其分为 3 类：

- MATLAB 内部函数，这是由 MATLAB 自身提供的，用户不能进行修改，这种函数如调试函数、快速傅立叶变换函数等。
- MATLAB 各种工具箱中的 M 文件提供了的大量的实用函数，这些函数在各自领域内具有广泛的用途。用户可以根据需要，对 M 文件内的语句作适当的修改与补充，以完成特定的功能。
- 用户自己编写的 M 文件。用户为了实现某种功能，或某一特定领域的算法，可以根据理论知识，自己进行编程。

  MATLAB 中提供的通用数理类函数包括：
  - 基本数学函数
  - 特殊函数
  - 基本矩阵函数
  - 特殊矩阵函数
  - 矩阵分解与分析函数
  - 数据分析函数
  - 微分方程求解
  - 非线性方程及其优化函数
  - 数值积分函数
  - 信号处理函数

6. 帮助命令

MATLAB 提供了非常方便的在线帮助命令(help)，它可提供各个函数的用法指南，包括格式、参数说明、注意事项及相关函数等。除此之外，help 命令还可提供有关 MATLAB 的一些重要信息。help 命令主要有以下几种格式。

- help

不带任何参数，这时将显示出 MATLAB 的目录项，产生类似于下列清单的信息：

```
help
HELP topics:
function\library     -  (No table of contents file)
book\ch4           -  (No table of contents file)
book\ch3           -  (No table of contents file)
```

```
book\chapter2          - (No table of contents file)
MATLAB\general         - General purpose commands.
MATLAB\ops             - Operators and special characters.
MATLAB\lang            - Programming language constructs.
MATLAB\elmat           - Elementary matrices and matrix manipulation.
MATLAB\elfun           - Elementary math functions.
MATLAB\specfun         - Specialized math functions.
MATLAB\matfun          - Matrix functions - numerical linear algebra.
MATLAB\datafun         - Data analysis and Fourier transforms.
MATLAB\polyfun         - Interpolation and polynomials.
MATLAB\funfun          - Function functions and ODE solvers.
MATLAB\sparfun         - Sparse matrices.
MATLAB\graph2d         - Two dimensional graphs.
MATLAB\graph3d         - Three dimensional graphs.
MATLAB\specgraph       - Specialized graphs.
MATLAB\graphics        - Handle Graphics.
MATLAB\uitools         - Graphical user interface tools.
MATLAB\strfun          - Character strings.
MATLAB\iofun           - File input/output.
MATLAB\timefun         - Time and dates.
MATLAB\datatypes       - Data types and structures.
MATLAB\winfun        - Windows Operating System Interface Files (DDE/ActiveX)
MATLAB\demos           - Examples and demonstrations.
toolbox\runtime        - MATLAB Runtime Server Development Kit
rtw\windows            - Real Time Windows Target.
daq\daq                - Data Acquisition Toolbox
daq\daqdemos           - Data Acquisition Toolbox - Data Acquisition Demos.
toolbox\dials            - Dials & Gauges Blockset
toolbox\rptgenext         - Simulink Report Generator
toolbox\rptgen           - MATLAB Report Generator
database\database         - Database Toolbox.
database\dbdemos         - Database Toolbox Demonstration Functions.
powersys\powerdemo     - Power System Blockset Demos.
powersys\powersys        - Power System Blockset
toolbox\compiler         - MATLAB Compiler (and Compiler 1.2.1)
comm\comm              - Communications Toolbox.
comm\commmasks         - Communications Toolbox mask helper functions.
comm\commsfun          - Communications Toolbox S-functions.
comm\commsim           - Communications Toolbox Simulink files.
toolbox\symbolic        - Symbolic Math Toolbox.
nag\nag             - NAG Foundation Toolbox - Numerical & Statistical Library
nag\examples          - NAG Foundation Toolbox - Numerical & Statistical Library
map\map                - Mapping Toolbox
map\mapdisp            - Mapping Toolbox Map Definition and Display.
map\mapproj            - Mapping Toolbox Projections.
wavelet\wavelet        - Wavelet Toolbox.
wavelet\wavedemo       - Wavelet Toolbox Demos.
toolbox\pde            - Partial Differential Equation Toolbox.
finance\finance        - Financial Toolbox.
finance\calendar       - Financial Toolbox calendar functions.
finance\findemos       - Financial Toolbox demonstration functions.
```

```
lmi\lmictrl          - LMI Control Toolbox: Control Applications
lmi\lmilab           - LMI Control Toolbox
qft\qft              - QFT Control Design Toolbox.
qft\qftdemos         - QFT Control Design Toolbox Demos
toolbox\fixpoint     - Fixed-Point Blockset
fixpoint\fxpdemos    - Fixed-Point Blockset Demos
fixpoint\obsolete    - Obsolete Fixed-Point Blockset
dspblks\dspblks      - DSP Blockset.
dspblks\dspmex       - (No table of contents file)
dspblks\dspdemos     - DSP Blockset demonstrations and examples.
dspblks\dspmasks     - DSP Blockset mask helper functions.
fuzzy\fuzzy          - Fuzzy Logic Toolbox.
fuzzy\fuzdemos       - Fuzzy Logic Toolbox Demos.
mpc\mpccmds          - Model Predictive Control Toolbox.
mpc\mpcdemos         - Model Predictive Control Toolbox
fdident\fdident      - Frequency Domain Identification Toolbox.
fdident\fddemos      - Demonstrations for the FDIDENT Toolbox
hosa\hosa            - Higher-Order Spectral Analysis Toolbox.
hosa\hosademo        - Higher-Order Spectral Analysis Toolbox - Demo suite
toolbox\stats        - Statistics Toolbox.
toolbox\ncd          - Nonlinear Control Design Blockset
images\images        - Image Processing Toolbox.
images\imdemos       - Image Processing Toolbox --- demos and sample images
nnet\nnet            - Neural Network Toolbox.
nnet\nndemos         - Neural Network Demonstrations.
nnet\nnutils         - (No table of contents file)
nnet\nnobsolete      - (No table of contents file)
mutools\commands     - Mu-Analysis and Synthesis Toolbox.
mutools\subs         - Mu-Analysis and Synthesis Toolbox.
signal\signal        - Signal Processing Toolbox.
signal\siggui        - Signal Processing Toolbox GUI
signal\sigdemos      - Signal Processing Toolbox Demonstrations
toolbox\splines      - Spline Toolbox.
toolbox\optim        - Optimization Toolbox.
toolbox\robust       - Robust Control Toolbox.
toolbox\ident        - System Identification Toolbox.
toolbox\control      - Control System Toolbox.
control\ctrlguis     - Control System Toolbox -- GUI support functions.
control\obsolete     - Control System Toolbox -- obsolete commands.
toolbox\rtw          - Real-Time Workshop
rtw\rtwdemos         - (No table of contents file)
stateflow\sfdemos    - Stateflow demonstrations and samples.
toolbox\sb2sl        - SystemBuild to Simulink Translator
stateflow\stateflow  - Stateflow
simulink\simulink    - Simulink
simulink\blocks      - Simulink block library.
simulink\simdemos    - Simulink 3 demonstrations and samples.
simulink\dee         - Differential Equation Editor
MATLAB\work          - (No table of contents file)
toolbox\local        - Preferences.
For more help on directory/topic, type "help topic".
```

在每一行中，显示出一个目录名及其对应目录的有关解释信息。

- help+目录名

  显示出指定目录中的所有命令及其函数。如输入：

```
help signal
```

则产生：

```
Signal Processing Toolbox.
  Version 5.1   (R12.1) 06-Apr-2001
   Filter analysis.
    abs          - Magnitude.
    angle        - Phase angle.
    filternorm   - Compute the 2-norm or inf-norm of a digital filter.
    freqs        - Laplace transform frequency response.
    freqspace    - Frequency spacing for frequency response.
    freqz        - Z-transform frequency response.
    freqzplot    - Plot frequency response data.
    fvtool       - Filter Visualization Tool (FVTool).
    grpdelay     - Group delay.
    impz         - Discrete impulse response.
    unwrap       - Unwrap phase.
    zplane       - Discrete pole-zero plot.
   Filter implementation.
    conv         - Convolution.
    conv2        - 2-D convolution.
    deconv       - Deconvolution.
    fftfilt      - Overlap-add filter implementation.
    filter       - Filter implementation.
    filter2      - Two-dimensional digital filtering.
    filtfilt     - Zero-phase version of filter.
    filtic       - Determine filter initial conditions.
    latcfilt     - Lattice filter implementation.
    medfilt1     - 1-Dimensional median filtering.
    sgolayfilt   - Savitzky-Golay filter implementation.
    sosfilt      - Second-order sections (biquad) filter implementation.
    upfirdn      - Up sample, FIR filter, down sample.
   FIR filter design.
    convmtx      - Convolution matrix.
    cremez       - Complex and nonlinear phase equiripple FIR filter design.
    fir1         - Window based FIR filter design - low, high, band, stop,
multi.
    fir2    - FIR arbitrary shape filter design using the frequency sampling
method.
    fircls       - Constrained Least Squares filter design - arbitrary
response.
    fircls1      - Constrained Least Squares FIR filter design - low and
highpass.
    firls        - Optimal least-squares FIR filter design.
    firrcos      - Raised cosine FIR filter design.
    intfilt      - Interpolation FIR filter design.
```

```
  kaiserord   - Kaiser window design based filter order estimation.
  remez       - Optimal Chebyshev-norm FIR filter design.
  remezord    - Remez design based filter order estimation.
  sgolay      - Savitzky-Golay FIR smoothing filter design.
 IIR digital filter design.
  butter      - Butterworth filter design.
  cheby1      - Chebyshev Type I filter design (passband ripple).
  cheby2      - Chebyshev Type II filter design (stopband ripple).
  ellip       - Elliptic filter design.
  maxflat     - Generalized Butterworth lowpass filter design.
  yulewalk    - Yule-Walker filter design.
 IIR filter order estimation.
  buttord     - Butterworth filter order estimation.
  cheb1ord    - Chebyshev Type I filter order estimation.
  cheb2ord    - Chebyshev Type II filter order estimation.
  ellipord    - Elliptic filter order estimation.
 Analog lowpass filter prototypes.
  besselap    - Bessel filter prototype.
  buttap      - Butterworth filter prototype.
  cheb1ap     - Chebyshev Type I filter prototype (passband ripple).
  cheb2ap     - Chebyshev Type II filter prototype (stopband ripple).
  ellipap     - Elliptic filter prototype.
 Analog filter design.
  besself     - Bessel analog filter design.
  butter      - Butterworth filter design.
  cheby1      - Chebyshev Type I filter design.
  cheby2      - Chebyshev Type II filter design.
  ellip       - Elliptic filter design.
 Analog filter transformation.
  lp2bp       - Lowpass to bandpass analog filter transformation.
  lp2bs       - Lowpass to bandstop analog filter transformation.
  lp2hp       - Lowpass to highpass analog filter transformation.
  lp2lp       - Lowpass to lowpass analog filter transformation.
 Filter discretization.
  bilinear    - Bilinear transformation with optional prewarping.
  impinvar    - Impulse invariance analog to digital conversion.
 Linear system transformations.
  latc2tf        -  Lattice  or  lattice  ladder  to  transfer  function
conversion.
  polystab    - Polynomial stabilization.
  polyscale   - Scale roots of polynomial.
  residuez    - Z-transform partial fraction expansion.
  sos2ss      - Second-order sections to state-space conversion.
  sos2tf      - Second-order sections to transfer function conversion.
  sos2zp      - Second-order sections to zero-pole conversion.
  ss2sos      - State-space to second-order sections conversion.
  ss2tf       - State-space to transfer function conversion.
  ss2zp       - State-space to zero-pole conversion.
  tf2latc        -  Transfer  function  to  lattice  or  lattice  ladder
conversion.
  tf2sos      - Transfer Function to second-order sections conversion.
```

```
    tf2ss       - Transfer function to state-space conversion.
    tf2zp       - Transfer function to zero-pole conversion.
    zp2sos      - Zero-pole to second-order sections conversion.
    zp2ss       - Zero-pole to state-space conversion.
    zp2tf       - Zero-pole to transfer function conversion.
  Windows.
    bartlett       - Bartlett window.
    barthannwin    - Modified Bartlett-Hanning window.
    blackman       - Blackman window.
    blackmanharris - Minimum 4-term Blackman-Harris window.
    bohmanwin      - Bohman window.
    chebwin        - Chebyshev window.
    gausswin       - Gaussian window.
    hamming        - Hamming window.
    hann           - Hann window.
    kaiser         - Kaiser window.
    nuttallwin       - Nuttall defined minimum 4-term Blackman-Harris
window.
    rectwin        - Rectangular window.
    triang         - Triangular window.
    tukeywin       - Tukey window.
    window         - Window function gateway.
  Transforms.
    bitrevorder - Permute input into bit-reversed order.
    czt         - Chirp-z transform.
    dct         - Discrete cosine transform.
    dftmtx      - Discrete Fourier transform matrix.
    fft         - Fast Fourier transform.
    fft2        - 2-D fast Fourier transform.
    fftshift    - Swap vector halves.
    goertzel    - Second-order Goertzel algorithm.
    hilbert     - Discrete-time analytic signal via Hilbert transform.
    idct        - Inverse discrete cosine transform.
    ifft        - Inverse fast Fourier transform.
    ifft2       - Inverse 2-D fast Fourier transform.
  Cepstral analysis.
    cceps       - Complex cepstrum.
    icceps      - Inverse Complex cepstrum.
    rceps       - Real cepstrum and minimum phase reconstruction.
  Statistical signal processing and spectral analysis.
    cohere      - Coherence function estimate.
    corrcoef    - Correlation coefficients.
    corrmtx     - Autocorrelation matrix.
    cov         - Covariance matrix.
    csd         - Cross Spectral Density.
    pburg       - Power Spectral Density estimate via Burg's method.
    pcov          - Power Spectral Density estimate via the Covariance
method.
    peig         - Power Spectral Density estimate via the Eigenvector
method.
    periodogram - Power Spectral Density estimate via the periodogram
```

```
method.
    pmcov - Power Spectral Density estimate via the Modified Covariance
method.
    pmtm   - Power Spectral Density estimate via the Thomson multitaper
method.
    pmusic      - Power Spectral Density estimate via the MUSIC method.
    psdplot     - Plot Power Spectral Density data.
    pwelch      - Power Spectral Density estimate via Welch's method.
    pyulear   - Power Spectral Density estimate via the Yule-Walker AR
Method.
    rooteig- Sinusoid frequency and power estimation via the eigenvector
algorithm.
    rootmusic - Sinusoid frequency and power estimation via the MUSIC
algorithm.
    tfe        - Transfer function estimate.
    xcorr      - Cross-correlation function.
    xcorr2     - 2-D cross-correlation.
    xcov       - Covariance function.
  Parametric modeling.
    arburg     - AR parametric modeling via Burg's method.
    arcov      - AR parametric modeling via covariance method.
    armcov     - AR parametric modeling via modified covariance method.
    aryule     - AR parametric modeling via the Yule-Walker method.
    ident      - See the System Identification Toolbox.
    invfreqs   - Analog filter fit to frequency response.
    invfreqz   - Discrete filter fit to frequency response.
    prony      - Prony's discrete filter fit to time response.
    stmcb      - Steiglitz-McBride iteration for ARMA modeling.
  Linear Prediction.
    ac2rc       - Autocorrelation sequence to reflection coefficients
conversion.
    ac2poly      - Autocorrelation sequence to prediction polynomial
conversion.
    is2rc       - Inverse sine parameters to reflection coefficients
conversion.
    lar2rc     - Log area ratios to reflection coefficients conversion.
    levinson   - Levinson-Durbin recursion.
    lpc        - Linear Predictive Coefficients using autocorrelation
method.
    lsf2poly    - Line spectral frequencies to prediction polynomial
conversion.
    poly2ac      - Prediction polynomial to autocorrelation sequence
conversion.
    poly2lsf    - Prediction polynomial to line spectral frequencies
conversion.
    poly2rc      - Prediction polynomial to reflection coefficients
conversion.
    rc2ac       - Reflection coefficients to autocorrelation sequence
conversion.
    rc2is       - Reflection coefficients to inverse sine parameters
conversion.
```

```
    rc2lar      - Reflection coefficients to log area ratios conversion.
    rc2poly      - Reflection coefficients to prediction polynomial
conversion.
    rlevinson   - Reverse Levinson-Durbin recursion.
    schurrc     - Schur algorithm.
   Multirate signal processing.
    decimate    - Resample data at a lower sample rate.
    downsample  - Downsample input signal.
    interp      - Resample data at a higher sample rate.
    interp1     - General 1-D interpolation. (MATLAB Toolbox)
    resample    - Resample sequence with new sampling rate.
    spline      - Cubic spline interpolation.
    upfirdn     - Up sample, FIR filter, down sample.
    upsample    - Upsample input signal.
   Waveform generation.
    chirp       - Swept-frequency cosine generator.
    diric       - Dirichlet (periodic sinc) function.
    gauspuls    - Gaussian RF pulse generator.
    gmonopuls   - Gaussian monopulse generator.
    pulstran    - Pulse train generator.
    rectpuls    - Sampled aperiodic rectangle generator.
    sawtooth    - Sawtooth function.
    sinc        - Sinc or sin(pi*x)/(pi*x) function
    square      - Square wave function.
    tripuls     - Sampled aperiodic triangle generator.
    vco         - Voltage controlled oscillator.
   Specialized operations.
    buffer      - Buffer a signal vector into a matrix of data frames.
    cell2sos    - Convert cell array to second-order-section matrix.
    cplxpair    - Order vector into complex conjugate pairs.
    demod       - Demodulation for communications simulation.
    dpss           - Discrete prolate spheroidal sequences (Slepian
sequences).
    dpssclear    - Remove discrete prolate spheroidal sequences from
database.
    dpssdir     - Discrete prolate spheroidal sequence database directory.
    dpssload      - Load discrete prolate spheroidal sequences from
database.
    dpsssave    - Save discrete prolate spheroidal sequences in database.
    eqtflength   - Equalize the length of a discrete-time transfer
function.
    modulate    - Modulation for communications simulation.
    seqperiod   - Find minimum-length repeating sequence in a vector.
    sos2cell    - Convert second-order-section matrix to cell array.
    specgram    - Spectrogram, for speech signals.
    stem        - Plot discrete data sequence.
    strips      - Strip plot.
    udecode     - Uniform decoding of the input.
    uencode      - Uniform quantization and encoding of the input into
N-bits.
   Graphical User Interfaces
```

```
      fdatool     - Filter Design and Analysis Tool.
      sptool      - Signal Processing Tool.
    See also SIGDEMOS, AUDIO, and, in the Filter Design Toolbox,
  FILTERDESIGN.
```

它包括信号处理工具箱的版本号、最新信息以及实现各种功能的函数及其说明。

- help+命令名/函数名/符号

显示出有关指定命令/函数/符号的详细信息，包括命令格式及注意事项。如输入：

```
help fft
```

将显示有关计算快速傅立叶变换的函数 fft 的说明：

```
FFT Discrete Fourier transform.
   FFT(X) is the discrete Fourier transform (DFT) of vector X.  If the
   length of X is a power of two, a fast radix-2 fast-Fourier
   transform algorithm is used.  If the length of X is not a
   power of two, a slower non-power-of-two algorithm is employed.
   For matrices, the FFT operation is applied to each column.
   For N-D arrays, the FFT operation operates on the first
   non-singleton dimension.
   FFT(X,N) is the N-point FFT, padded with zeros if X has less
   than N points and truncated if it has more.
   FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
   dimension DIM.
   For length N input vector x, the DFT is a length N vector X,
   with elements
            N
   X(k) = sum x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
            n=1
   The inverse DFT (computed by IFFT) is given by
            N
     x(n) = (1/N) sum X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
            k=1
   The relationship between the DFT and the Fourier coefficients a and b
in
            N/2
   x(n) = a0 + sum a(k) *cos(2*pi*k*t(n)/(N*dt))+b(k) *sin(2*pi*k*t(n)/(N*
dt))
            k=1
            is
     a0 = X(1)/N, a(k) = 2*real(X(k+1))/N, b(k) = -2*imag(X(k+1))/N,
   where x is a length N discrete signal sampled at times t with spacing
dt.
See also IFFT, FFT2, IFFT2, FFTSHIFT.
```

1.2.2 矩阵运算

矩阵运算是 MATLAB 的基础，所有参与运算的数都被看作为矩阵，如常数被看作为 1×1 矩阵，向量被看作为 $1\times n$ 或 $n\times1$ 矩阵。

1. 矩阵的加法与减法运算

矩阵与矩阵相加或相减，就是把这两个矩阵中所对应的元素相加或相减，并且这两个矩阵必须维数相同。如下例所示：

```
a=randn(3,4)
a =
   -0.4326    0.2877    1.1892    0.1746
   -1.6656   -1.1465   -0.0376   -0.1867
    0.1253    1.1909    0.3273    0.7258
b=randn(3,4)
b =
   -0.5883    0.1139   -0.0956   -1.3362
    2.1832    1.0668   -0.8323    0.7143
   -0.1364    0.0593    0.2944    1.6236
```

则

```
c=a+b
c =
   -1.0209    0.4016    1.0935   -1.1615
    0.5176   -0.0797   -0.8700    0.5276
   -0.0111    1.2502    0.6217    2.3494
d=a-b
d =
     0.1557    0.1737    1.2848    1.5108
    -3.8488   -2.2132    0.7947   -0.9010
     0.2617    1.1316    0.0329   -0.8978
```

另外，如果矩阵与标量进行加减运算，则矩阵中的每个元素都与标量进行加减运算。例如：

```
x=[1 2 3;4 5 6];
y=x-1
```

则有

```
y =
     0     1     2
     3     4     5
```

2. 矩阵转置

实矩阵的转置操作，就是将矩阵中的元素 a_{ij} 与元素 a_{ji} 互换，例如：

```
a=[4 2 1;3 4 5;6 7 8]
a =
     4     2     1
     3     4     5
     6     7     8
b=a'
b =
```

```
     4     3     6
     2     4     7
     1     5     8
```

如果矩阵 A 为复数矩阵，则 A'表示矩阵 A 的共轭矩阵，而 A.'才为矩阵 A 的转置，例如：

```
a=[3 2;4 5]+i*[5 6;8 7]
a =
   3.0000 + 5.0000i   2.0000 + 6.0000i
   4.0000 + 8.0000i   5.0000 + 7.0000i
b=a'
b =
   3.0000 - 5.0000i   4.0000 - 8.0000i
   2.0000 - 6.0000i   5.0000 - 7.0000i
c=a.'
c =
   3.0000 + 5.0000i   4.0000 + 8.0000i
   2.0000 + 6.0000i   5.0000 + 7.0000i
```

3. 矩阵乘法

MATLAB 使用"*"作为矩阵相乘的运算符。矩阵乘法不满足交换律，也就是说如果矩阵乘积 A*B 与 B*A 都有意义，则它们的值一般也不相等，例如：

```
a=[1 3 4;2 5 6;7 8 9]
a =
     1     3     4
     2     5     6
     7     8     9
b=[3 2 1;5 3 6;2 3 1]
b =
     3     2     1
     5     3     6
     2     3     1
c=a*b
c =
    26    23    23
    43    37    38
    79    65    64
d=b*a
d =
    14    27    33
    53    78    92
    15    29    35
```

矩阵与标量相乘，即为矩阵中的各元素分别与标量相乘，例如：

```
a=[1 3 4;2 5 6;7 8 9]
a =
     1     3     4
     2     5     6
```

```
     7     8     9
b=a*2
b =
     2     6     8
     4    10    12
    14    16    18
```

对于维数相同的两个向量相乘，即为两个向量的内积，例如：

```
x=[1 -2 1]';
y=[3 4 5]';
z=x'*y
```

则有

```
z =
     0
```

4. 矩阵除法

MATLAB 中提供了两种除法运算：左除(\)与右除(/)，如果 a 为一非奇异矩阵，则 a\b 与 b/a 可通过 a 的逆阵与 b 阵得到：

```
a\b=inv(a)*b
b/a=b*inv(a)
```

一般情况下，x=a\b 是方程 a*x=b 的解，而 x=b/a 是方程 x*a=b 的解。

例如：

```
a=[3 1;2 4];
b=[4 5];
c=a/b
```

则得：

```
c =
     0.4146
     6829
```

5. 矩阵行列式与范数

MATLAB 中函数 det 用来计算矩阵的行列式。求矩阵的范数使用函数 norm(x,p)，其中 x 为给定的矩阵，p 为范数类型，通常取值为 1、2、inf，默认值为 2。

例如：

```
a=[1 2 3;4 3 6;7 5 9];
b=det(a)
c=norm(a)
```

则得：

```
b =
    6
```

```
c =
    15.1152
```

1.2.3　矩阵分解

MATLAB 中共有 4 大类矩阵分解函数：三角分解、正交分解、奇异值分解与特征值分解。

1.　三角分解

三角分解是将矩阵分解成两个基本三角阵的乘积，其中一个为上三角阵，另一个为下三角阵。这种分解通常称为 LU 分解，其算法大多为高斯变量消元法的变型。

MATLAB 中利用函数 lu 实现矩阵的三角分解。例如：

```
a=[3 2 4;5 2 6;1 8 9];
[l,u]=lu(a)
```

得到：

```
l =
    0.6000    0.1053    1.0000
    1.0000         0         0
    0.2000    1.0000         0
u =
    5.0000    2.0000    6.0000
         0    7.6000    7.8000
         0         0   -0.4211
```

可见 l 与 u 分别为下三角阵和上三角阵。同样可以利用 l * u 来验证分解的正确性。

```
a=l*u
a =
    3.0000    2.0000    4.0000
    5.0000    2.0000    6.0000
    1.0000    8.0000    9.0000
```

LU 分解可用于简化逆矩阵的求逆与求行列式的值。例如求 a 的逆阵时，除了可以通过函数 inv 来实现，也可用 LU 分解来实现。

输入：

```
x=inv(a)
x1=inv(u)*inv(l)
```

得到：

```
x =
        1.8750   -0.8750   -0.2500
        2.4375   -1.4375   -0.1250
       -2.3750    1.3750    0.2500
x1 =
        1.8750   -0.8750   -0.2500
        2.4375   -1.4375   -0.1250
```

```
    -2.3750   1.3750   0.2500
```

对于小矩阵来说，这两种方法没有任何区别。但当矩阵的维数相当大时，利用函数 inv 计算矩阵的逆速度非常慢，会大大影响算法的效率。如果采用 LU 分解，将原矩阵置换为特殊的下三角阵与上三角阵，再计算矩阵的逆就会非常快捷。

同样，利用 LU 分解可计算矩阵 a 的行列式，输入：

```
b=det(a)
c=det(l)*det(u)
```

得到

```
b =
   -16.0000
c =
   -16.0000
```

2. 正交分解

正交分解(QR)可将方阵和长方矩阵分解成一个正交矩阵与一个上三角阵的乘积。MATLAB 中的函数 qr 可以实现正交分解。

例如，输入

```
a=[3 2 1;6 5 4;9 8 7];
[q,r]=qr(a)
```

得到

```
q =
   -0.2673    0.8729    0.4082
   -0.5345    0.2182   -0.8165
   -0.8018   -0.4364    0.4082
r =
   -11.2250   -9.6214   -8.0178
         0    -0.6547   -1.3093
         0          0    0.0000
```

3. 奇异值分解

在 MATLAB 中，奇异值分解可通过下列语句来实现：

```
[u, s, v]=svd(a)
```

其中 svd 为 MATLAB 中的奇异值分解函数，u 和 v 为正交矩阵，s 为对角阵，它们之间的关系为：

```
a=u*s*v'
```

如果要求函数 svd(a)只返回一个参数，则返回对角阵 s，即为 a 的奇异值。

奇异值分解在矩阵分析中占有非常重要的地位。例如：

```
a=[3 2 1;6 5 4;9 8 7];
[u,s,v]=svd(a)
```

得到

```
u =
    0.2148   -0.8872    0.4082
    0.5206   -0.2496   -0.8165
    0.8263    0.3879    0.4082
s =
    16.8481       0          0
       0       1.0684        0
       0          0       0.0000
v =
    0.6651   -0.6253    0.4082
    0.5724    0.0757   -0.8165
    0.4797    0.7767    0.4082
```

4. 特征值分解

对于 n×n 的矩阵 a，我们把满足 ax=λx 的λ值称为 a 的特征值，x 称为特征向量。MATLAB 中的函数 eig(a)可以得到特征值构成的列向量。如果 a 为实对称矩阵，那么特征值为实数；如果 a 为非对称矩阵，则特征值通常为复数。

例如：

```
a=[3 2 1;6 5 4;9 8 7];
eig(a)
ans =
       13.6847
        1.3153
       -0.0000
```

利用下面的格式：

```
[x,d]=eig(a)
```

可以同时得到矩阵 a 的特征值与特征矩阵。其中，d 阵对角线上的元素为特征值，而 x 矩阵的列为相应的特征向量。

1.2.4 数据分析与统计

MATLAB 为了进行数据统计分析，提供了大量的数据分析函数，以方便用户调用。

1. 面向列的数据分析

MATLAB 习惯用矩阵来表示数据，其中同一列数值表示一个变量，每一行则对应于各个变量的一组值，这也是 MATLAB 中的数据分析函数通常对应列操作的原因所在。

MATLAB 提供了大量的数据分析常用函数，如表 1.3 所示。

例如，有一数据文件的存放目录为 d:\seaclutter\seadata\data1.dat，利用 load 命令可调出数据：

```
load d:\seaclutter\seadata\data1.dat
```

数据内容为：

```
data1 =
    0.3899    0.1286   -0.1199
    0.0880    0.6565   -0.0653
   -0.6355   -1.1678    0.4853
   -0.5596   -0.4606   -0.5955
    0.4437   -0.2624   -0.1497
   -0.9499   -1.2132   -0.4348
    0.7812   -1.3194   -0.0793
    0.5690    0.9312    1.5352
   -0.8217    0.0112   -0.6065
   -0.2656   -0.6451   -1.3474
   -1.1878    0.8057    0.4694
   -2.2023    0.2316   -0.9036
    0.9863   -0.9898    0.0359
   -0.5186    1.3396   -0.6275
    0.3274    0.2895    0.5354
    0.2341    1.4789    0.5529
    0.0215    1.1380   -0.2037
   -1.0039   -0.6841   -2.0543
   -0.9471   -1.2919    0.1326
   -0.3744   -0.0729    1.5929
   -1.1859   -0.3306    1.0184
   -1.0559   -0.8436   -1.5804
    1.4725    0.4978   -0.0787
    0.0557    1.4885   -0.6817
   -1.2173   -0.5465   -1.0246
   -0.0412   -0.8468   -1.2344
   -1.1283   -0.2463    0.2888
   -1.3493    0.6630   -0.4293
   -0.2611   -0.8542    0.0558
    0.9535   -1.2013   -0.3679
```

表 1.3 数据分析常用函数

函 数 名	功能描述
Cumprod	元素累积积
Cumsum	元素累积和
Cumtrapz	梯形积分累计
Max	求列最大值
Mean	求列平均值
Median	求列中值
Min	求列最小值
Perms	求所有可能的排列
Prod	各元素的积
Sort	各列元素升序排列

续表

函数名	功能描述
Sortrows	按行升序排列
Std	求列标准差
Sum	求列和
Trapz	求梯形数值积分

利用上面的数据分析常用函数可以分析此数据的均值、标准差、列最大值及列最小值等。

输入下面的语句：

```
juzhi=mean(data1)
bzc=std(data1)
zdz=max(data1)
zxz=min(data1)
```

所得结果为：

```
juzhi =
        -0.3128   -0.1105   -0.1961
bzc =
         0.8444    0.8785    0.8313
zdz =
         1.4725    1.4885    1.5929
zxz =
        -2.2023   -1.3194   -2.0543
```

由此可见，这些数据分析函数是按列对数据进行处理的。如果要计算数据的误差矩阵，可以通过下面的语句实现。

```
[row,col]=size(data1);
I=ones(row,1);
Error=data1-I*juzhi
```

所得的误差矩阵为：

```
Error =
    0.7026    0.2392     0.0762
    0.4008    0.7670     0.1308
   -0.3227   -1.0573     0.6814
   -0.2468   -0.3501    -0.3994
    0.7564   -0.1519     0.0464
   -0.6371   -1.1026    -0.2387
    1.0939   -1.2089     0.1167
    0.8817    1.0418     1.7312
   -0.5089    0.1218    -0.4104
    0.0472   -0.5346    -1.1513
   -0.8750    0.9163     0.6654
   -1.8896    0.3422    -0.7075
    1.2991   -0.8792     0.2319
```

```
   -0.2059    1.4501    -0.4315
    0.6401    0.4000     0.7315
    0.5468    1.5895     0.7489
    0.3342    1.2486    -0.0076
   -0.6912   -0.5736    -1.8583
   -0.6344   -1.1814     0.3286
   -0.0617    0.0376     1.7890
   -0.8731   -0.2201     1.2145
   -0.7431   -0.7331    -1.3843
    1.7852    0.6083     0.1174
    0.3685    1.5990    -0.4856
   -0.9046   -0.4359    -0.8285
    0.2715   -0.7362    -1.0383
   -0.8156   -0.1358     0.4849
   -1.0365    0.7736    -0.2332
    0.0517   -0.7436     0.2519
    1.2662   -1.0908    -0.1718
```

2. 数据预处理

在 MATLAB 中，当遇到没有意义的表达式，如 0/0 与 inf/inf 时，计算结果中会出现一个特殊的变量 NaN。在对数据进行统计分析前，可以先将数据中含有的 NaN 删除。表 1.4 给出了删除数据中 NaN 的几个函数。

表 1.4　从数组中删除 NaN 的函数

i=find(~isNaN(x)) x=x(i)	在矢量中查找所有不是 NaN 的元素，然后保存不是 NaN 的元素
x=x(find(~isNaN(x)))	从矢量中删除 NaN
x=x(~isNaN(x))	从矢量中删除 NaN，速度更快
x(isNaN(x))=[]	从矢量中删除 NaN
X(any(isNaN(X)'),:) =[]	删除矩阵 X 中包含 NaN 的行

例如，产生一 4×4 的随机矩阵，并令其中某些项为 NaN，利用表 1.4 中的方法删除矩阵中包含 NaN 的行。

输入下列语句：

```
data=randn(4);
data(2,2)=NaN;
data(3,4)=NaN
```

结果为：

```
data =
     -0.4650    -1.3573    -1.3813    1.9574
      0.3710     NaN        0.3155    0.5045
      0.7283     1.0378     1.5532    NaN
      2.1122    -0.3898     0.7079   -0.3398
```

再输入语句：

```
data(any(isnan(data))',:)=[]
```

删除包含 NaN 的行后的结果为：

```
data =
     -1.1398    0.6353    0.0860   -0.3210
     -1.1162   -1.0998    0.4620   -2.3252
```

3. 协方差矩阵与相关系数矩阵

MATLAB 中的函数 cov 用于计算数组的协方差矩阵。对于 m×n 维的数组，其协方差矩阵为 n 阶方阵。这里计算数据 data1.dat 的协方差矩阵。

输入语句：

```
xfcm=cov(data1)
```

结果为：

```
xfcm =
        0.7130    0.0446    0.1750
        0.0446    0.7717    0.1621
        0.1750    0.1621    0.6910
```

从中可以看出，协方差矩阵为对称矩阵。

相关系数是衡量两个矢量线性关系紧密程度的量，其值介于 0 与 1 之间，若两矢量相等，则相关系数为 1，若两矢量相互独立，则相关系数为 0。在 MATLAB 中，函数 corrcoef 用于计算矢量组的相关系数矩阵。对于矩阵来说，相关系数矩阵为各列矢量的相关系数。

对于数据 data1.dat，输入下面的语句计算其相关系数矩阵：

```
xg=corrcoef(data1)
```

输出结果为：

```
xg =
      1.0000    0.0601    0.2493
      0.0601    1.0000    0.2220
      0.2493    0.2220    1.0000
```

4. 曲线拟合

曲线拟合的方法有很多种，其中比较常见的有两种：多项式拟合与指数拟合。由于曲线拟合问题最终归结为线性方程组的求解，这在 MATLAB 中正好对应于矩阵的求逆或除法运算，所以很容易得到解决。这里以二次多项式拟合为例，来说明如何采用 MATLAB 实现曲线拟合，并由此推出指数拟合的实现方法。

假设对应于时间矢量 $t=[t_1, t_2, t_3, t_4]$，测得一组矢量 $x=[x_1, x_2, x_3, x_4]$，若采用二次多项式拟合数据曲线，即

$x=a_0+a_1t+a_2t^2$

则将矢量 t 与 x 代入拟合方程得到线性方程组：

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

解这个线性方程组可得未知系数 a_0、a_1、a_2。

根据上述理论，二次多项式拟合可以通过下面的 MATLAB 语句实现：

```
T=[ones(size(t))  t  t.^2];
a=T \ x
```

同理，对于带有线性参数的指数拟合

$x=a_0+a_1e^{-t}+a_2te^{-t}$

其 MATLAB 实现语句为：

```
T=[ones(size(t))  exp(-t)  t.*exp(-t)];
a=T \ x
```

另外，MATLAB 工具箱中的 polyfit 函数可以自动完成上述功能。

1.3 绘图功能

MATLAB 6 中含有丰富的图形绘制函数，包括二维图形绘制、三维图形绘制以及通用绘图工具函数等，同时还包括一些专业绘图函数，如绘制条形图、箭形图及等高线图等，因而其具有强大的绘图功能。

1.3.1 基本的绘图功能

1. 绘制简单的曲线

绘制简单的二维曲线可以利用函数 plot 来实现，而函数 plot3 则用来绘制简单的三维线图。

- 二维绘图函数 plot

  函数 plot 根据输入参数的不同，可以分为以下几种格式：

  - plot(y)

    当 y 为一向量时，它以 y 的序号作为 X 轴，按向量 y 的值绘制曲线。
  - plot(x,y)

    x、y 均为向量时，这时以向量 x 作为 X 轴，向量 y 作为 Y 轴，绘制出典型的二维曲线。
  - plot(x,y1,s1,x,y2,s2,…)

    以公共的 x 向量作为 X 轴，分别以 y1，y2，…绘制出多条曲线，每条曲线的外形可由相应的字符 s1，s2，…来指定，包括曲线的颜色与字体。指定颜色的字符见表 1.5，指定线型的字符见表 1.6。

表 1.5 基本绘图函数的颜色设置

符 号	表示的颜色
Y	黄色
G	绿色
B	蓝色
M	红紫色
C	蓝绿色
W	白色
R	红色
K	黑色

表 1.6 基本绘图函数的线型设置

符 号	表示的线型
+	加号形状
O	空心圆状
*	星号
.	实心小点形状
X	叉号形状
S	方形
D	菱形
^	向上箭头
V	向下箭头
>	向右箭头
<	向左箭头
P	五角星形
H	六角星形

例如在二维平面内绘制下面的 3 条曲线：

```
y=cos(x)
y1=cos(x+0.25)
y2=cos(x+0.5)
```

可以通过下面的简单语句实现：

```
x=0:pi/100:2*pi;
y=cos(x);
y1=cos(x+0.25);
y2=cos(x+0.5);
plot(x,y,'b-',x,y1,'g.',x,y2,'r--')
```

得到如图 1.13 所示的曲线。

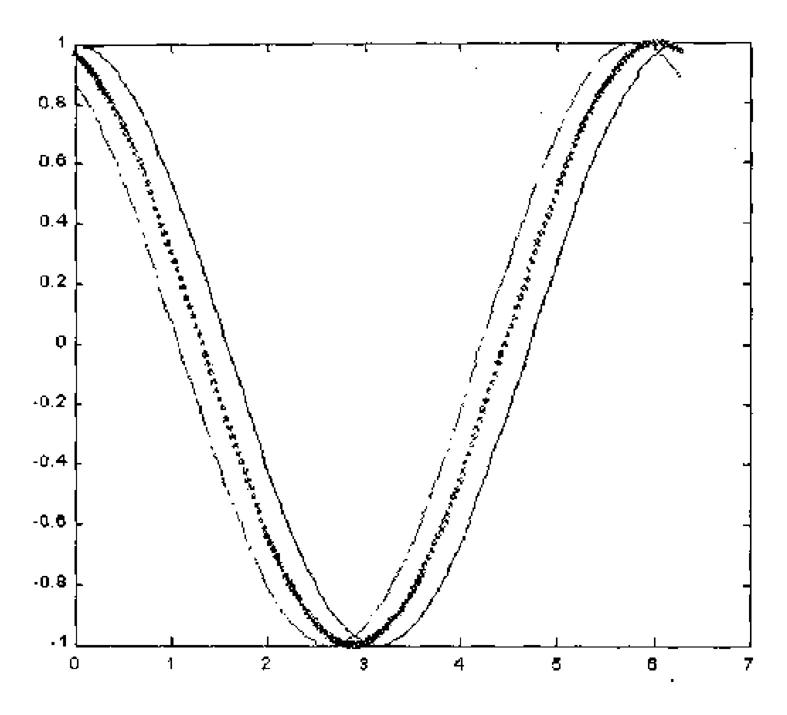


图1.13 在二维平面上绘制多条曲线

- 三维绘图函数 plot3

  对三维绘图函数 plot3，有两种格式：

  ◆ plot(x，y，z)

  如果 x、y、z 为 3 个长度相等的向量，则在三维空间中产生一条曲线，每一点的坐标相对于向量 x、y、z 的值。

  例如用下面的语句产生一个三维螺旋线。

```
t=0:pi/50:5*pi;
x=cos(t);
y=sin(t);
plot3(t,x,y)
```

  如果 x、y、z 为相同大小的矩阵，则可在三维空间中绘制出由 x、y、z 的列构成的一组曲线。

  例如，输入语句

```
[X,Y]=meshgrid([-3:0.2:3]);
Z=X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z)
grid on
```

  得到的三维螺旋线如图 1.14 所示。

  ◆ plot3(x，y，z，cs)

  变量 x、y、z 的用法如同格式 1。参数 cs 用于指定曲线的颜色与线型格式，其字符说明见表 1.5 与表 1.6。

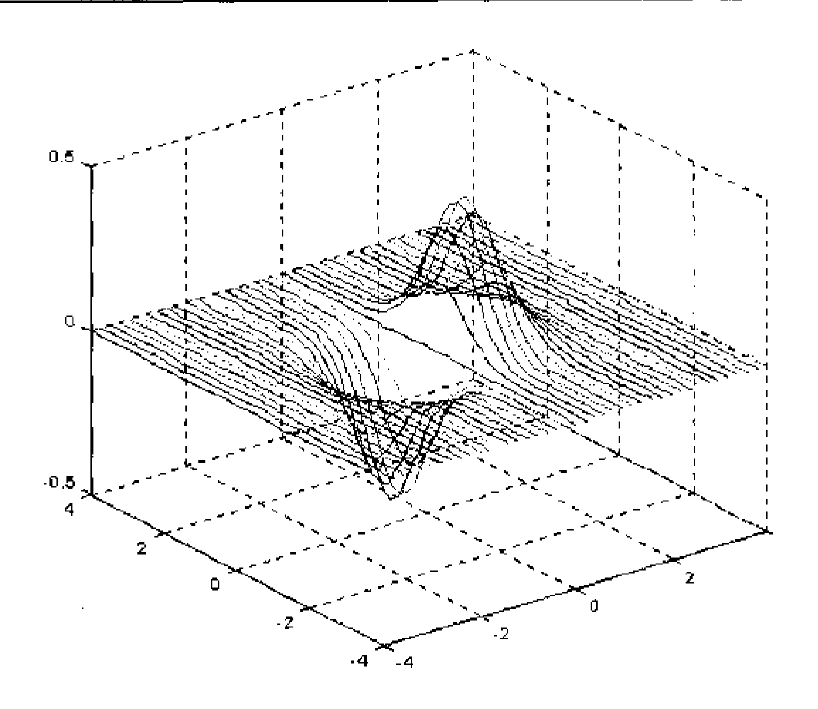


图1.14 绘制三维螺旋线

2. 设置坐标轴的参数

在创建图形时，MATLAB 自动选择坐标轴表示的数值范围，并用一定的数据间隔标记坐标轴的数据。用户也可以自己指定坐标的范围与数据间隔。下面的几个命令与设置坐标轴的参数有关：

- axis，控制坐标轴的刻度和形式。

  格式 AXIS([XMIN XMAX YMIN YMAX])用于设置当前图形的 x 轴与 y 轴范围。

  格式 AXIS([XMIN XMAX YMIN YMAX ZMIN ZMAX])用于指定三维图形的 x 轴、y 轴及 z 轴的范围。

  如果只指定坐标轴的最大值或最小值，其他坐标轴的范围用 inf 或 -inf 表示，则产生半自动坐标轴范围。例如：

```
x=0:pi/100:2*pi;
y=cos(x).^2;
plot(x,y)
axis([0 inf 0 1])
```

  得到如图 1.15 所示的图形。

  另外，在默认情况下，直角坐标图形的纵横比与图形窗口的纵横比相同，使用函数 axis 可以控制图形的纵横比，以更适宜的方式显示图形窗口中的图形。函数 axis 控制图形纵横比的格式有 3 种：

  - axis square：将两个轴的长度设置为相等；
  - axis equal：将坐标轴的标记间距设置为相等；
  - axis equal tight：将图形以紧缩方式显示。

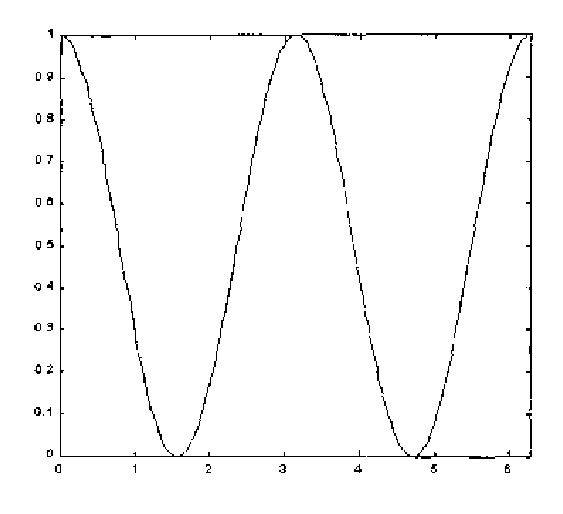


图1.15 设置坐标轴的范围

以上面的例子为例，3 种坐标轴控制方式得到的图形分别如图 1.16~1.18 所示。

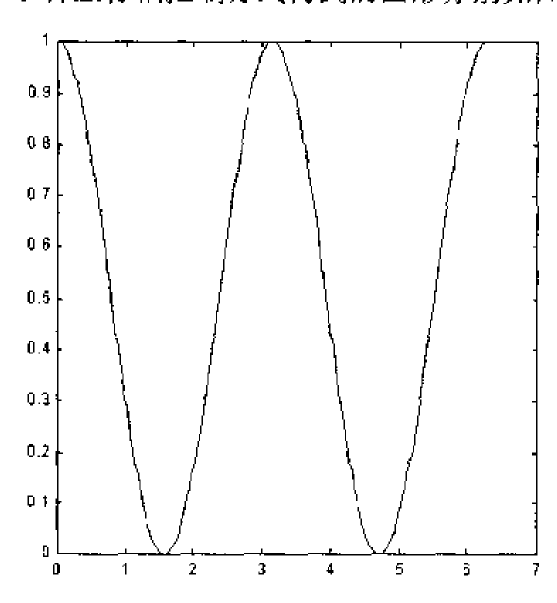


图1.16 两个坐标轴的长度相等时的图形

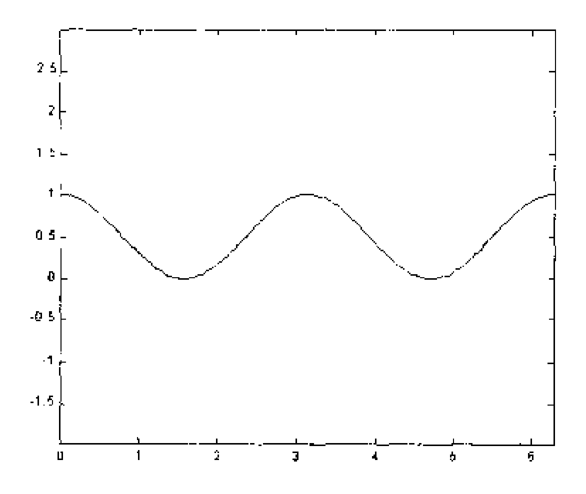


图1.17 坐标轴标记间距相等时的图形

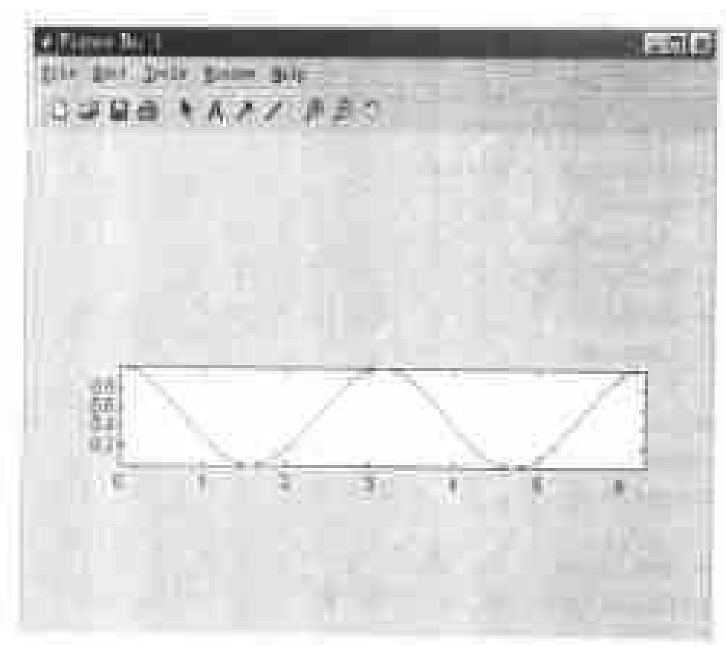


图1.18 以紧缩方式显示时的图形

● axes，用指定的特征创建一个新坐标轴。

格式 AXES('position', RECT)用于在指定的位置给出坐标轴并返回它的句柄。

格式 RECT = [left, bottom, width, height]中，参数 left 与 bottom 指定坐标轴相对于当前归一化窗口的位置，(0, 0)代表图形的左上角，(1, 1)代表图形的右上角。参数 width 与 height 指定坐标轴的宽度和高度。

● gca，返回当前轴的句柄。

格式 H=GCA 返回当前图形中坐标轴的句柄，这在获取或设置轴的各种特性时经常用到。

● get 和 set，获取或设置已有轴的各种特性。

get 函数用于获取已有轴的各种特性。例如，对于如图 1.18 所示的图形，若输入以下语句：

```
H=gca;
Get(H)
```

则返回该图形的各种特性如下：

```
AmbientLightColor = [1 1 1]
    Box = on
    CameraPosition = [3.14159 0.5 33.3461]
    CameraPositionMode = auto
    CameraTarget = [3.14159 0.5 0]
    CameraTargetMode = auto
    CameraUpVector = [0 1 0]
    CameraUpVectorMode = auto
    CameraViewAngle = [10.7641]
    CameraViewAngleMode = auto
    CLim = [0 1]
    CLimMode = auto
```

```
Color = [1 1 1]
CurrentPoint = [ (2 by 3) double array]
ColorOrder = [ (7 by 3) double array]
DataAspectRatio = [1 1 1]
DataAspectRatioMode = manual
DrawMode = normal
FontAngle = normal
FontName = Helvetica
FontSize = [10]
FontUnits = points
FontWeight = normal
GridLineStyle = :
Layer = bottom
LineStyleOrder = -
LineWidth = [0.5]
NextPlot = replace
PlotBoxAspectRatio = [6.28319 1 2]
PlotBoxAspectRatioMode = manual
Projection = orthographic
Position = [0.13 0.11 0.775 0.815]
TickLength = [0.01 0.025]
TickDir = in
TickDirMode = auto
Title = [77.0002]
Units = normalized
View = [0 90]
XColor = [0 0 0]
XDir = normal
XGrid = off
XLabel = [74.0007]
XAxisLocation = bottom
XLim = [0 6.28319]
XLimMode = manual
XScale = linear
XTick = [ (1 by 7) double array]
XTickLabel =
    0
    1
    2
    3
    4
    5
    6
XTickLabelMode = auto
XTickMode = auto
YColor = [0 0 0]
YDir = normal
YGrid = off
YLabel = [75.0002]
YAxisLocation = left
YLim = [2.58606e-032 1]
```

```
YLimMode = manual
YScale = linear
YTick = [0.2 0.4 0.6 0.8]
YTickLabel =
    0.2
    0.4
    0.6
    0.8
YTickLabelMode = auto
YTickMode = auto
ZColor = [0 0 0]
ZDir = normal
ZGrid = off
ZLabel = [76.0002]
ZLim = [-1 1]
ZLimMode = auto
ZScale = linear
ZTick = [-1 0 1]
ZTickLabel =
ZTickLabelMode = auto
ZTickMode = auto
ButtonDownFcn =
Children = [1.00098]
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [1]
Selected = off
SelectionHighlight = on
Tag =
Type = axes
UIContextMenu = []
UserData = []
Visible = on
```

对于线性坐标轴，MATLAB 6 根据数据的范围自动设置等间距的坐标轴数值标记。用户可以利用函数 set，通过修改句柄变量 gca 中的某些特性，来自定义坐标轴，包括数值标记以及在坐标轴上标记字符等。

利用 set 函数修改 gca 变量中的 xtick 或 ylick 的属性值，自定义坐标轴的标记。例如，输入下面语句：

```
x=0:pi/100:2*pi;
y=cos(x).^2;
plot(x,y)
set(gca,'ytick',[0:0.05:1])
```

得到的图形如图 1.19 所示。

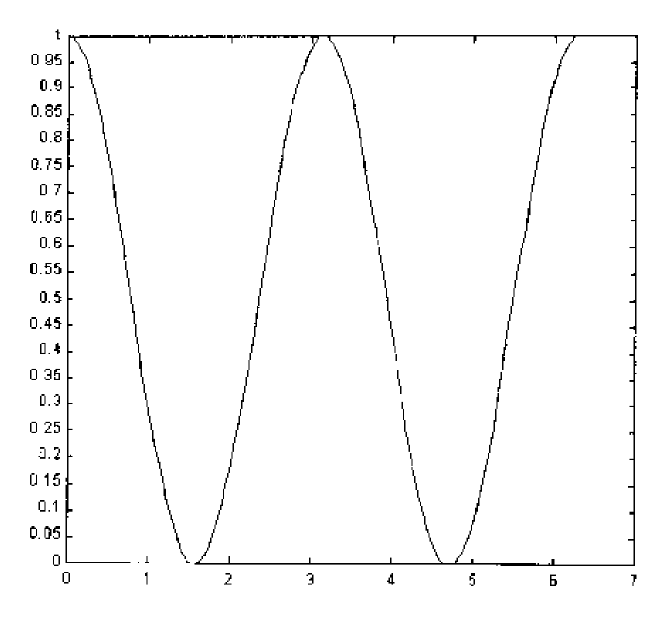


图1.19 设置坐标轴的数值标记

利用 set 函数修改 gca 变量中的 xticklabel 的属性值，在坐标轴上标记字符。例如，输入下面语句：

```
x=0:pi/100:2*pi;
y=cos(x).^2;
plot(x,y)
set(gca,'xtick',[0:pi/2:2*pi])
set(gca,'xticklabel',{'0','pi/2','pi','3pi/2','2pi'})
```

得到的图形如图 1.20 所示。

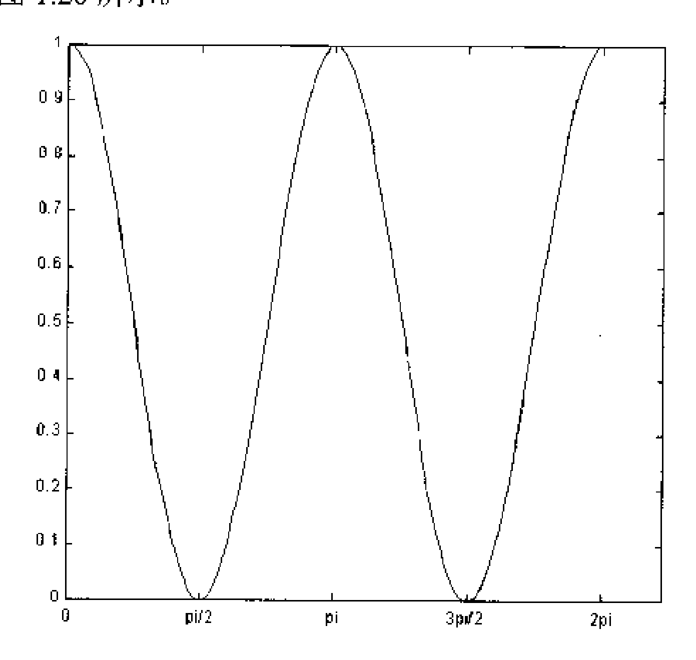


图1.20 用字符标记X轴

3. 图形标注

MATLAB 6 提供了几个函数用于对图形进行标注，不仅可以对坐标轴进行标注，也可以在图形中添加标题，或在图形中的任意位置放置标注，还可以显示图例。表 1.7 给出了常用的标注函数及其功能描述，以便用户参考。

表 1.7 常用的图形标注函数及其功能描述

函 数 名	功能描述
Gtext	使用鼠标在图形的任意位置放置标注
Legend	在图形中添加图例
Text	在指定位置显示文本
Title	添加图形标题
Xlabel	添加 X 轴标注
Ylabel	添加 Y 轴标注
Zlabel	添加 Z 轴标注

- 坐标轴标注与添加图形标题

使用函数 xlabel、ylabel 与 zlabel 分别给坐标轴添加标注，使用函数 title 给图形添加标题。通过下面的例子说明这 3 个函数的功能。

输入下面语句：

```
x=0:pi/100:2*pi;
y=cos(x);
plot(x,y)
xlabel('x 的范围为[0 2\pi]','fontsize',10)
ylabel('y=cos(x)','fontsize',10)
title('余弦函数曲线')
```

得到的图形如图 1.21 所示。

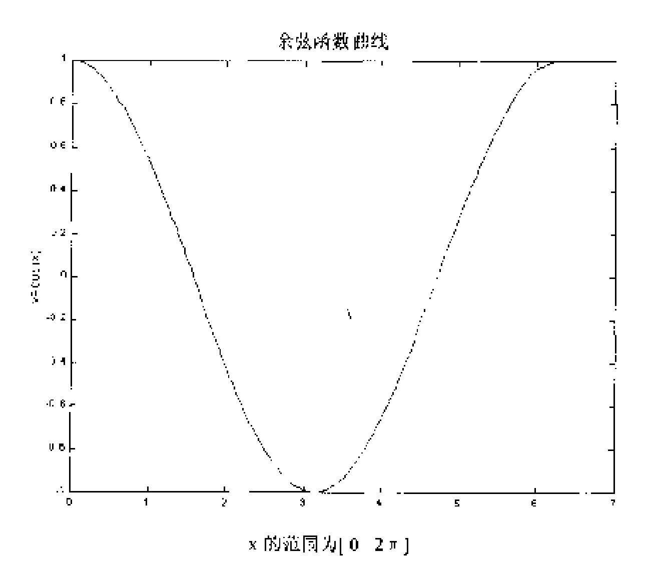


图1.21 坐标轴标注与添加标题

- 在图形中添加文本与图例

使用函数 text 可将文本字符串放置在图形中的任意位置，放置字符串的位置可以用图形上的点表示。

例如，使用下面的语句可以为上例的图形添加标注：

```
x=0:pi/100:2*pi;
y=cos(x);
plot(x,y)
xlabel('x 的范围为[0 2\pi]','fontsize',10)
ylabel('y=cos(x)','fontsize',10)
title('余弦函数曲线')
    text(3*pi/4,cos(3*pi/4),'\rightarrowcos(3\pi/4)',...
       'fontsize',10)
text(7*pi/4,cos(7*pi/4),'\leftarrowcos(7\pi/4)',...
'fontsize',10)
```

标注后的图形如图 1.22 所示。

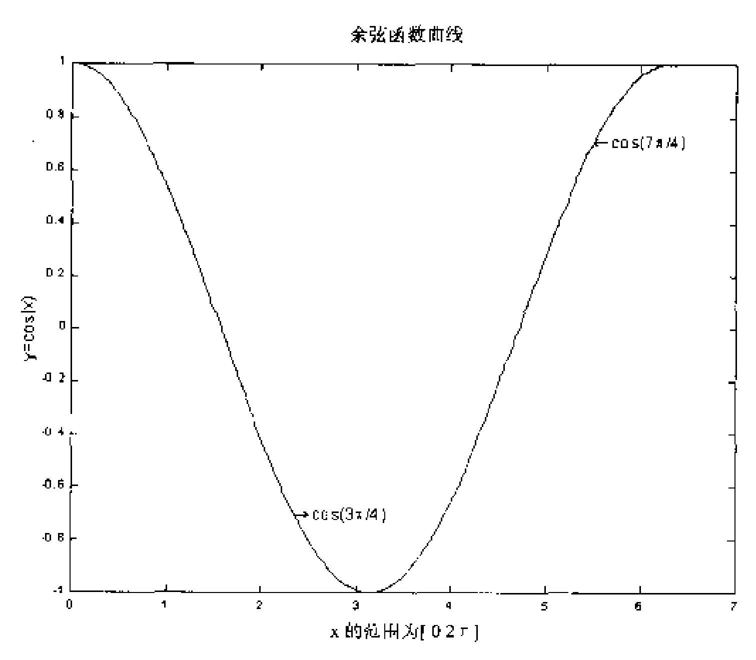


图1.22　图形的文本标注

使用函数 legned 可以在图形中添加图例，不同图形的线型及颜色自动生成。

例如，输入下面语句：

```
t=0:pi/100:2*pi;
y=cos(t);
z=sin(t);
plot(x,y,'b-',x,z,'g.')
h=legend('cos','sin');
```

显示图例结果如图 1.23 所示。

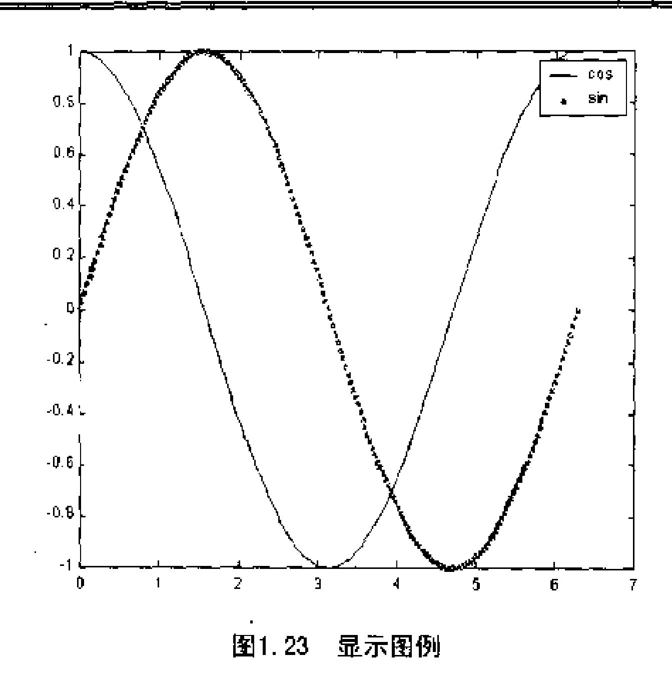


图1.23 显示图例

1.3.2 专业绘图功能

MATLAB 6 中提供了多个用于专业图形的绘制的函数，如绘制条形图、饼图、三维饼图、箭头图、星点图、阶梯图及等高线等。

1. 条形图

条形图用于显示矢量或矩阵中的数据，适合显示离散数据，并能比较不同数据组的结果。绘制条形图的函数主要有以下几种：

- 函数 bar 用于绘制垂直条形图；
- 函数 bar3 用于绘制三维垂直条形图；
- 函数 barh 用于绘制水平条形图；
- 函数 bar3h 用于绘制三维水平条形图。

对于 m×n 维的矩阵，函数 bar 绘制得到 m 组条形图，每组有 n 个垂直的条形图。例如对于 4×5 维的矩阵 a，利用函数 bar 得到其条形图如图 1.24 所示。

```
a=[1 2 3 2 1;2 3 4 3 2;3 4 5 4 3;4 5 6 5 4];
bar(a)
legend('第一列','第二列','第三列','第四列','第五列',2)
grid
```

从图中可以看出，共绘制了 5 组条形图，每组图与矩阵的行相对应，而每组图又由 5 个条形图组成，分别对应每一行中的各列，且每一列元素被自动分配一种颜色。

若利用函数 barh 创建矩阵 a 的水平条形图，则输入下面语句：

```
barh(a)
legend('第一列','第二列','第三列','第四列','第五列',4)
grid
```

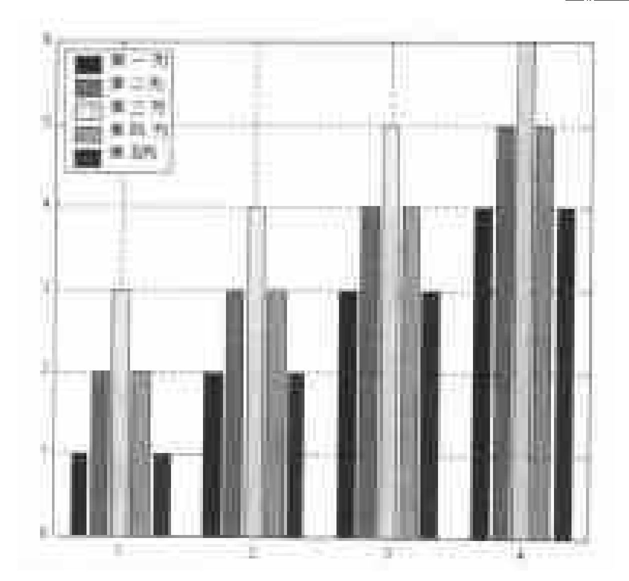


图1.24 矩阵的垂直条形图

得到水平条形图如图 1.25 所示。

另外，如果指定函数 bar 的输入参数'stack'，则可以将每一组条形图连接成一个条形图，形成重叠式条形图。

例如对矩阵 a，输入下面的语句：

```
bar(a,'stack')
legend('第一列','第二列','第三列','第四列','第五列',2)
grid
```

得到重叠式条形图如图 1.26 所示。

三维条形图最简单的形式是根据矩阵中每个元素绘制成一个长方体。各行元素沿 X 轴分布，各列元素沿 Y 轴分布，即用 X 轴、Y 轴表示元素的位置，用 Z 轴表示元素的大小。

对于矩阵 a，输入：

```
bar3(a)
xlabel('x');
ylabel('y');
zlabel('z');
legend('第一列','第二列','第三列','第四列','第五列',1)
```

得到三维条形图如图 1.27 所示。

上面绘制的三维条形图都是表示矩阵元素相互分离的长方体，如果在函数 bar3 中添加参数，就可获得如图 1.28 所示的三维分组式条形图，它是以每行的元素作为一组，例如第 2 组表示的是矩阵中第 2 行的 5 个元素。

```
bar3(a,'group')
xlabel('x'); ylabel('y'); zlabel('z');
legend('第一列','第二列','第三列','第四列','第五列',1)
```

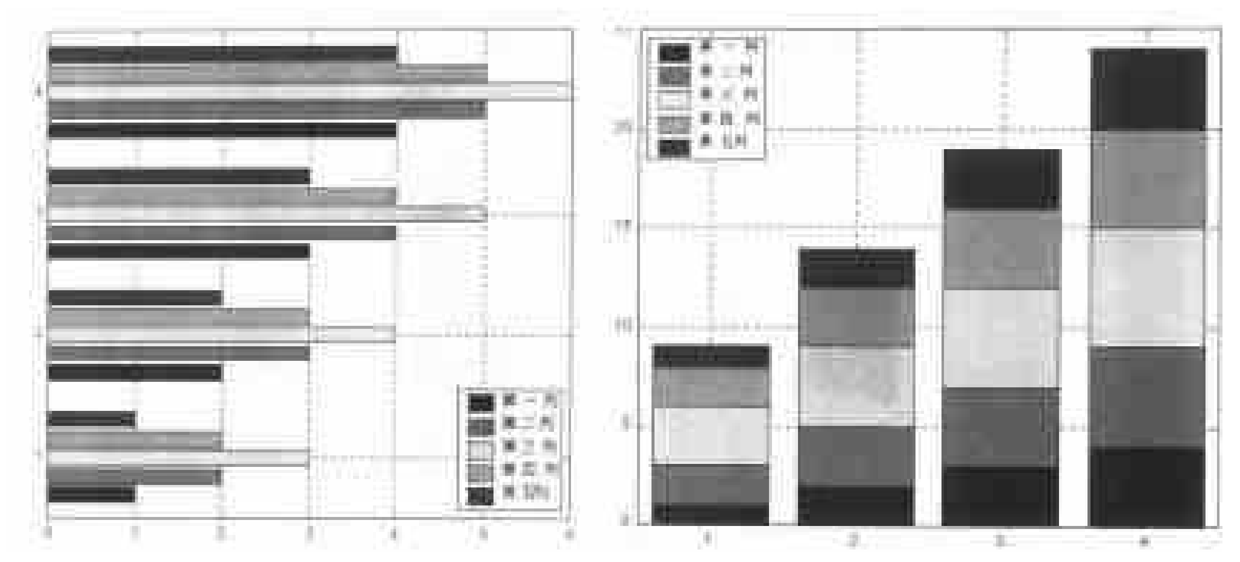


图1.25　矩阵的水平条形图　　　　图1.26　重叠式条形图

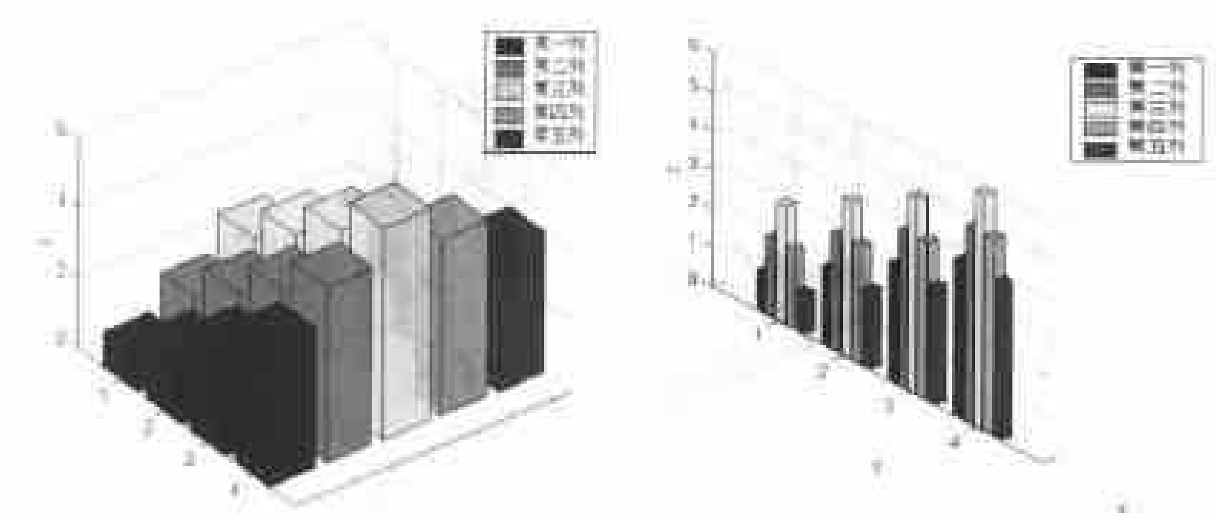


图1.27　三维条形图　　　　图1.28　三维分组式条形图

类似于二维条形图函数，可以得到矩阵 a 的三维条形图的水平结构与重叠结构。可以通过下述语句实现：

```
figure(1)
subplot(1,2,1)
bar3h(a)
title('一般式');
subplot(1,2,2)
bar3h(a,'group')
title('分组式');
figure(2)
subplot(1,2,1)
bar3(a,'stack')
title('垂直');
subplot(1,2,2)
bar3h(a,'stack')
title('水平')
```

得到的图形如图 1.29 与 1.30 所示。

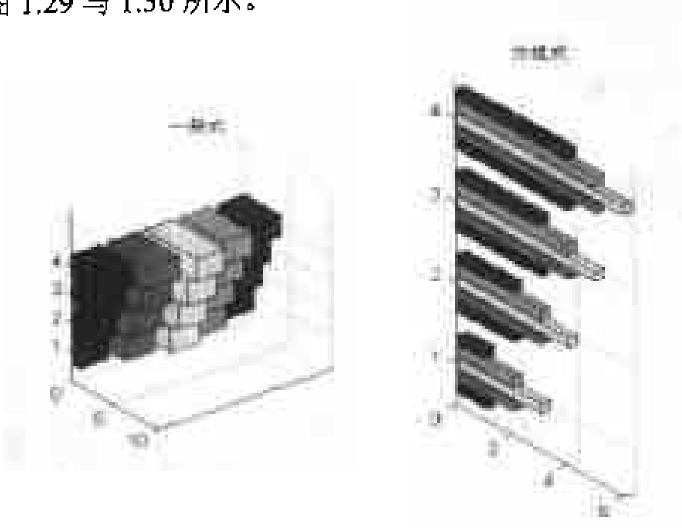


图1.29 水平方向的三维条形图

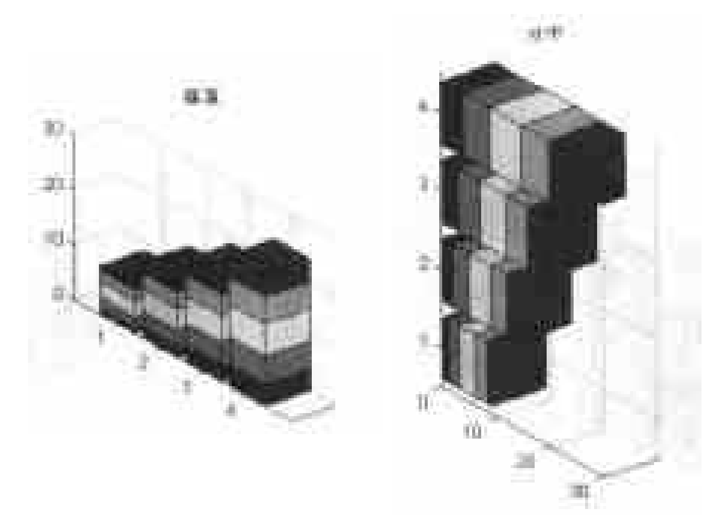


图1.30 三维重叠式条形图

2. 绘制离散数据的图形

MATLAB 6 中提供的绘制离散数据图形的函数有：stem(杆图)、stem3(三维杆图)与stairs(阶梯图)。使用函数 stem 和 stem3 绘制离散数据时，各点之间不用线连接，而是每个点对应于一条与横轴垂直的线段，线段的上端点可以指定不同的样式。

下面用一个例子来说明函数 stem 的用法，并通过该例比较杆图与线图的区别。输入下面的语句：

```
n=0:pi/20:2*pi;
xn=sin(n);
subplot(1,2,1)
plot(n,xn)
xlabel('n');ylabel('x(n)');
title('线图')
grid
```

```
subplot(1,2,2)
stem(n,xn)
xlabel('n');ylabel('x(n)');
title('杆图')
grid
```

得到的线图与杆图如图 1.31 所示。

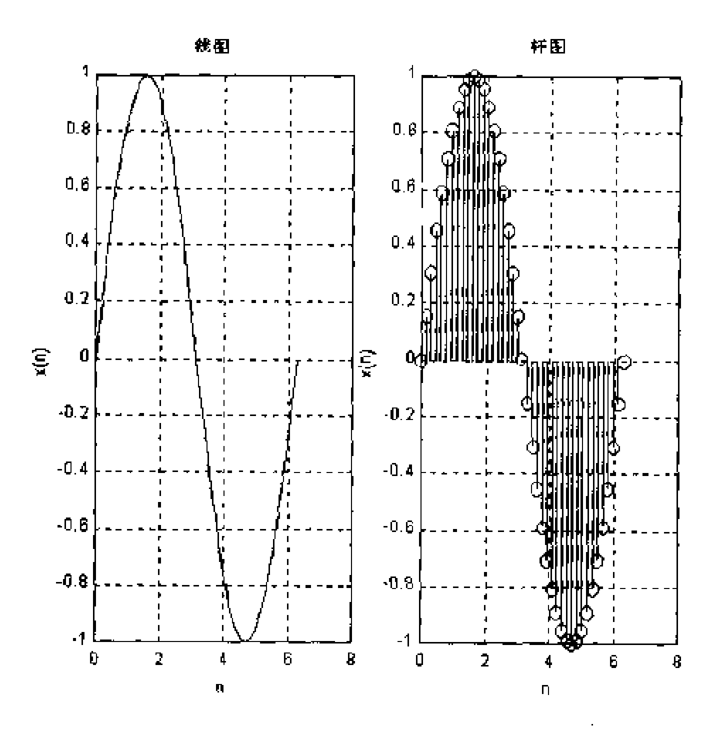


图1.31　线图与杆图的比较

从图中可以看出，用 stem 函数绘制的杆图与用 plot 函数绘制的线图相似，只是杆图是一些离散的线段，而线图是连续的图形。

有时为了对同一组数据的不同形式进行比较，需要在同一图形中同时绘制杆图与线图。例如，输入下面的语句：

```
n=0:pi/20:2*pi;
xn=sin(n);
yn=cos(n);
stem_handles=stem(n,xn+yn);
hold on
plot_handles=plot(n,xn,'g-',n,yn,'r.')
legend_handles=[stem_handles(1);plot_handles];
legend(legend_handles,'xn+yn','xn=sin(n)','yn=cos(n)')
xlabel('n');
ylabel('amplitude');
hold off
```

得到的图形如图 1.32 所示。

函数 stem3 用于绘制三维杆图，该函数专为不能用二维方式可视化的数据设计，例如，对复平面上单位圆上的点进行快速傅立叶变换时，可用函数 stem3 将快速傅立叶变换后的结果可视化。通过下面的程序可以实现如图 1.33 所示的三维杆图。

```
nfft=256;
n=2*pi*(0:nfft-1)/nfft;
xn=cos(n);
yn=sin(n);
h=randn(1,100);
f=fft(h,nfft);
g=abs(f);
stem3(xn,yn,g,'c','fill');
view([-65 45])
xlabel('real');
ylabel('imaginary');
zlabel('amplitude');
title('amplitude frequency response');
```

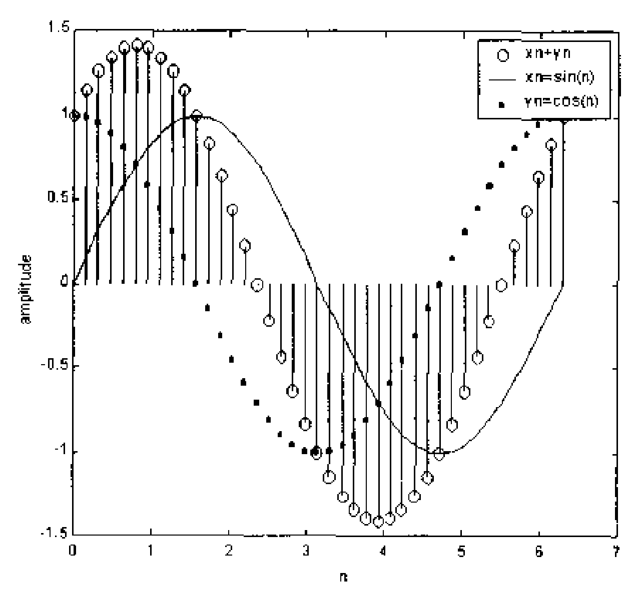


图1.32 杆图与线图的组合图

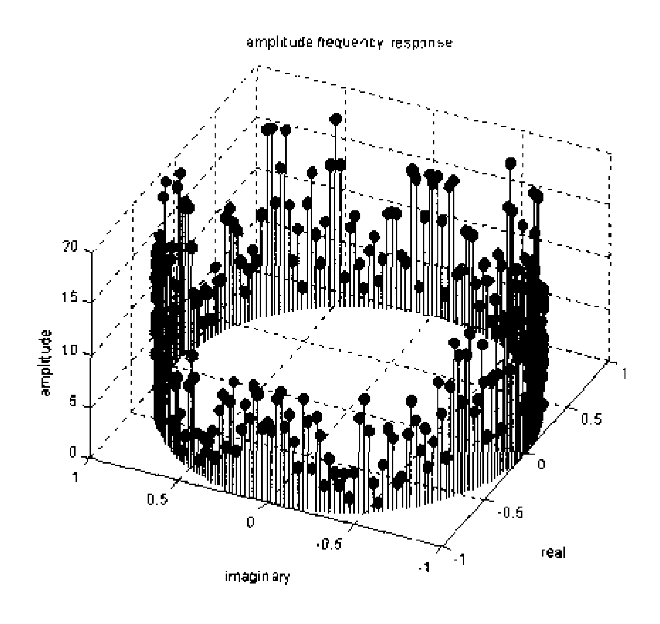


图1.33 三维杆图

类似于二维杆图与线图的组合，有时也将三维杆图与三维线图进行组合，例如当进行拉普拉斯变换时，其基本函数 e^{-st} 就可用函数 stem3 绘制的三维杆图与函数 plot3 绘制的三维线图相组合而可视化。

例如，输入下面的语句，得到如图 1.34 所示的三维杆图与三维线图的组合图。

```
t=0:0.1:10;
s=0.01+0.9i;
x=exp(-s*t);
stem3(real(x),imag(x),t);
hold on
plot3(real(x),imag(x),t,'r');
hold off
xlabel('real');
ylabel('imaginary');
zlabel('amplitude');
```

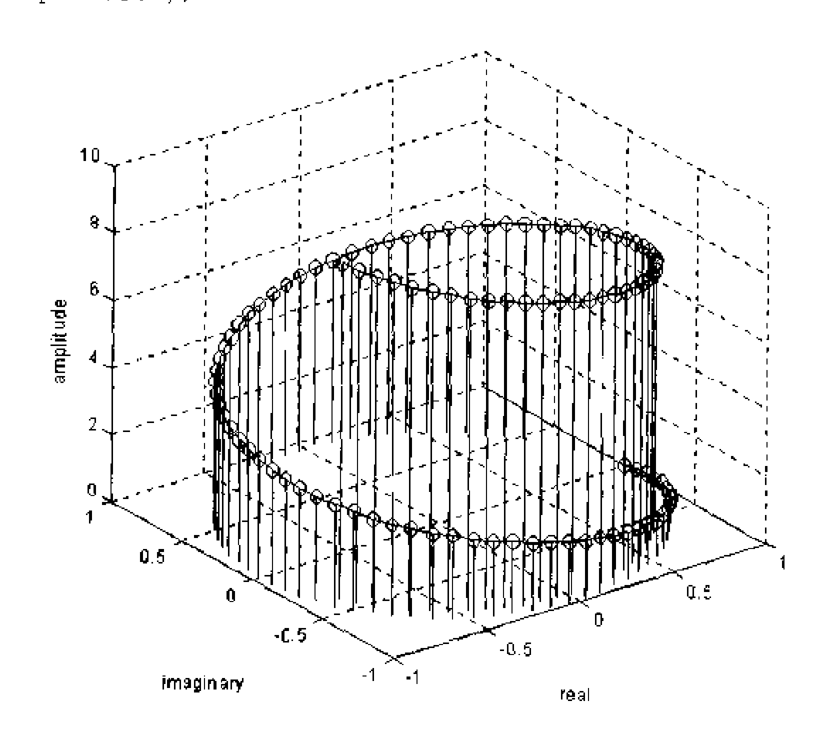


图1.34 三维杆图与三维线图的组合图

要绘制数字采样与时间的关系图，可以利用函数 stairs 来实现。例如用下面的语句可以实现一组数据的梯形图与线图，并对图形进行标注，结果如图 1.35 所示。

```
n=0:20;
x=(0.9).^n;
stairs(n,x)
grid
hold on
plot(n,x,'r')
hold off
label='0.9^n 的梯形图';
title(label,'fontsize',10);
```

```
xlabel('n');
ylabel('x(n)');
```

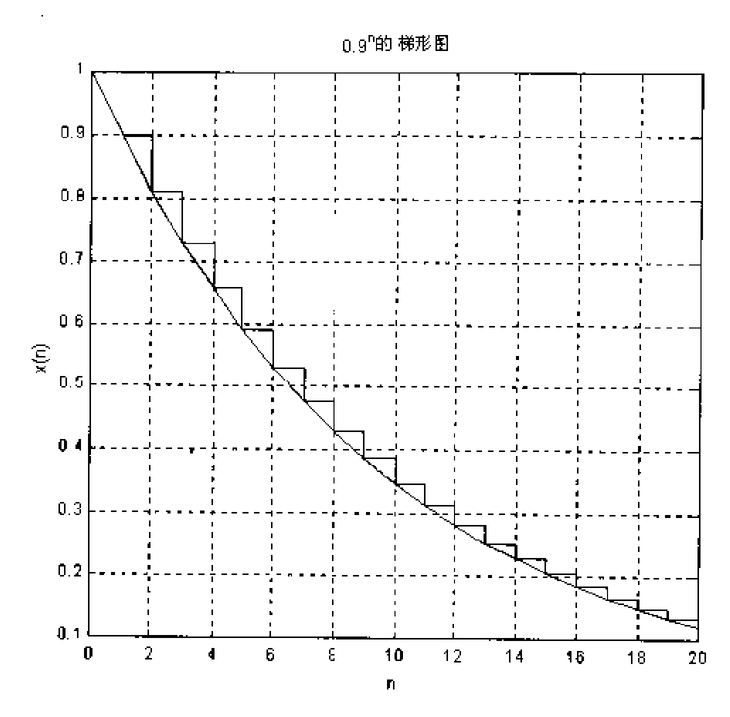


图1.35 梯形图与线图的结合

1.4 M文件

当用户实现一些简单功能如简单的计算与绘图时，因为输入的语句不多，可以在命令窗口中一行一行地输入，并能方便地修改。但如果实现较复杂的功能，且需要经常修改其中的参数或多次调用，这时在命令窗口中一行一行输入，显然不能满足要求，因此我们必须利用 MATLAB 语言编写驱动文件，即 M 文件，其扩展名为.m。例如，besttree.m 就是由一系列 MATLAB 语句编写的 M 文件，它可用来产生小波包分解的最佳树。

用户可以通过文本编辑器或字处理器生成 M 文件，且 M 文件之间可以相互调用，用户可以根据自己的需要，自我编写 M 文件。从功能上分，M 文件可分为底稿文件与函数文件两类。调用底稿文件会自动执行一系列命令直至给出结果；函数文件则为 MATLAB 提供了扩充性，通过函数文件，可以产生完成某一特定功能的函数。

1.4.1 底稿文件

底稿文件由一系列 MATLAB 语句组成。当调用一个底稿文件时，MATLAB 自动执行文件中的一系列语句，在执行过程中，并不交互地等待键盘的输入。底稿文件中的语句可使用工作空间中的全局变量，因此，它在实现分析问题、解决问题及设计复杂命令等方面是十分有用的。

底稿文件是一种简单的 M 文件，它没有输入、输出参数。例如下面是一个简单的底稿文件，用于绘制一个花瓣图案。

```
%用于创建花瓣图形的底稿文件 petrals.m
%采用极坐标方式显示图形
theta=0:pi/100:2*pi;
pet(1,:)=2*sin(3*theta).^2;
pet(2,:)=cos(6*theta).^3;
pet(3,:)=sin(2*theta).^2;
pet(4,:)=3*cos(4*theta).^3;
for i=1:4
polar(theta,pet(i,:));
end
```

其中“%”右边的语句为说明语句，它们只起到注释或帮助的作用。在 M 文件编辑器中输入上述语句，并保存为 petals.m，则文件 petals.m 就是一个底稿文件。在 MATLAB 命令窗口中输入 petals，则 MATLAB 会自动执行这一文件中的每条命令，并得到如图 1.36 所示的花瓣图形。

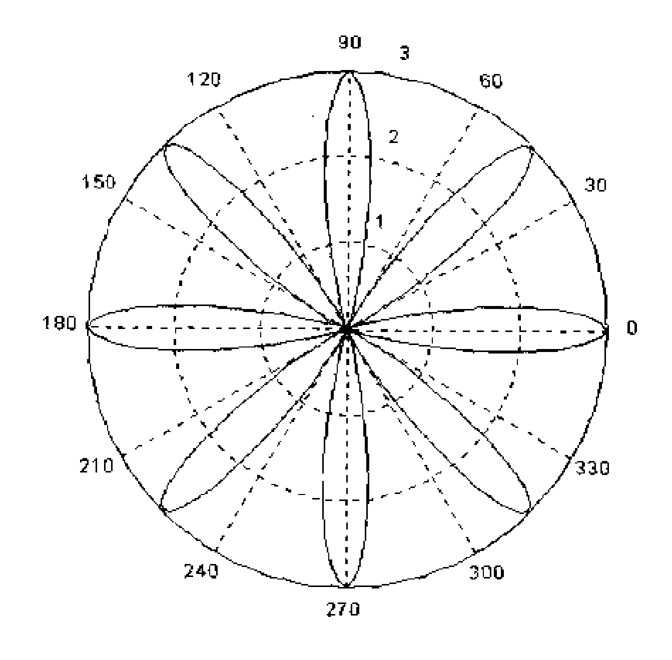


图1.36　底稿文件petals.m的运行结果

程序执行后工作空间中的变量如图 1.37 所示。底稿文件所创建的工作空间变量 i、pet、theta 都可以继续使用。

MATLAB 提供了许多关于如何使用底稿文件解决复杂问题的演示示例，用户可以通过在命令窗口中输入 Demos 调用这些示例。

在加载 MATLAB 时，系统会自动执行底稿文件 startup.m，这个文件类似于 DOS 系统中的 autoexe.bat 文件。因此用户可以把要在工作空间中预定义的特征常数、工作系统等信息加到该文件中。这样在系统启动时就可自动执行，而无需在每次启动后重新设置。在多用户或网络环境中，底稿文件 matlabrc.m 是专为系统管理员使用的，通过它可完成系统的定义及信息传递。

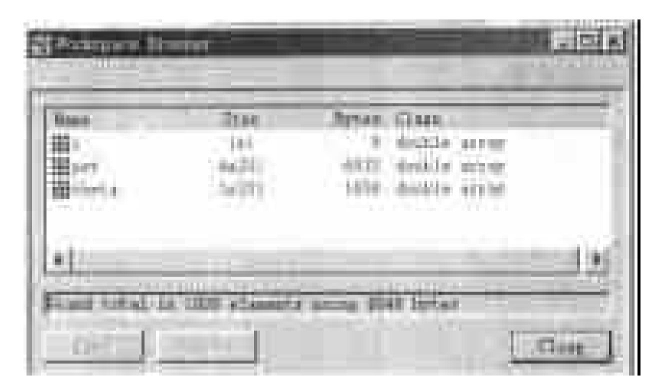


图1.37 底稿文件petals.m创建的变量

1.4.2 函数文件

函数文件的第 1 行必须包含关键字 “function”。函数文件与底稿文件的区别在于：函数文件可以接受输入参数，并可返回输出参数，而底稿文件不具备参数传递功能；在函数文件中定义及使用的变量大都是局部变量，只在本函数的工作区内有效，一旦退出该函数，即为无效变量，而底稿文件中定义或使用的变量都是全局变量，在退出文件后仍为有效变量。

1. 函数文件的结构

下面的函数 mean.m 是一个简单的 M 文件，用于计算矢量中元素的平均值。

```
function y=mean(x)
%该程序用于计算矢量中元素的平均值
%x 为输入参数，代表输入的矢量
%y 为函数返回参数，代表矢量中元素的平均值
%如果输入不是向量，返回错误信息
[m,n]=size(x);
%判断输入是否为矢量
if(~((m==1)|(n==1))|(m==1&n==1))
%如果输入不是矢量，显示出错信息
error('输入必须是向量');
end
y=sum(x)/length(x);
```

将上面的代码编辑成一个 M 文件，并保存为 mean.m。这个函数接受一个参数，并返回一个输出参数。若在命令窗口中输入下面语句：

```
s=1:20;
m=mean(s)
```

则调用 mean 函数的返回结果为：

```
m =
   10.5000
```

从上面的函数文件可以看出，一个函数文件通常由 4 个部分组成：函数定义、帮助文

本、函数体与注释。

- 函数定义

函数定义说明在 M 文件中包含一个函数，并确定调用的参数及其次序。例如函数文件 mean.m 的函数定义行为：

```
function y=mean(x)
```

其中 function 为用于定义函数的关键字，y 为输出参数，mean 为函数名，x 为输入参数。所有 MATLAB 函数都必须有函数定义行，如果有多个输出参数，可以用一对方括号包括所有输出参数列表，并用逗号分隔，返回输出参数的次序与此参数列表的次序相同。如果有多个输入参数，则按顺序用逗号分隔，所有输入参数都放在一对圆括号中。例如：

```
[Pxx,Pxxc,f]=psd(x,nfft,Fs,window,noverlap,p)
```

- 帮助文本

在函数定义行与函数体之间的注释语句为帮助文本。通常将函数定义后的第 1 行注释语句称为帮助文本的首行。首行的特殊性在于可以用命令 lookfor 搜索其中的字符，因此通常在首行中添加关于函数的摘要。用 help 命令可以显示帮助文本。通常在帮助文本中输入关于函数使用方法的信息，如各参数的意义、函数文件的功能及相关函数。例如，函数文件 mean.m 的帮助文本为：

```
%该程序用于计算矢量中元素的平均值
%x 为输入参数，代表输入的矢量
%y 为函数返回参数，代表矢量中元素的平均值
%如果输入不是向量，返回错误信息
```

- 函数体

函数体包含所有用于计算和为输出参数赋值的语句行。在函数体中的语句可以是赋值表达式、计算表达式、函数调用、流程控制等程序结构、交互式的输入与输出、注释等。例如，函数文件 mean.m 的函数体为：

```
[m,n]=size(x);
%判断输入是否为矢量
if(~((m==1)|(n==1))|(m==1&n==1))
%如果输入不是矢量，显示出错信息
error('输入必须是向量');
end
y=sum(x)/length(x);
```

其中第 7 行调用函数 sum 与 length，第 3 行到第 6 行为一个流程控制语句，第 5 行显示出错信息，第 2 行与第 4 行为注释。

- 注释

注释以“%”开头，可以出现在函数体中的任意一行，可以单独占一行，也可以放在语句之后。例如，函数文件 mean.m 的注释为：

```
%判断输入是否为矢量
%如果输入不是矢量，显示出错信息
```

在函数体中的注释行不能用 help 命令查看。

2. 检查函数文件的参数个数

使用函数 nargin 和 nargout 可以确定函数被调用时，使用输入参数和输出参数的个数，然后可以用条件表达式根据参数的数目应用不同的处理方法。

例如，MATLAB 信号处理工具箱中的函数文件 psd.m:

```
function [Pxx, Pxxc, f] = psd(varargin)
%PSD Power Spectral Density estimate.
error(nargchk(1,7,nargin))
x = varargin{1};
[msg,nfft,Fs,window,noverlap,p,dflag]=psdchk(varargin(2:end),x);
error(msg)
% compute PSD
window = window(:);
n = length(x);          % Number of data points
nwind = length(window); % length of window
if n < nwind     % zero-pad x if it has length less than the window length
   x(nwind)=0;  n=nwind;
end
% Make sure x is a column vector; do this AFTER the zero-padding
% in case x is a scalar.
x = x(:);

k = fix((n-noverlap)/(nwind-noverlap));    % Number of windows
                                    % (k = fix(n/nwind) for noverlap=0)
Spec = zeros(nfft,1); Spec2 = zeros(nfft,1);
for i=1:k
   if strcmp(dflag,'none')
      xw = window.*(x(index));
   elseif strcmp(dflag,'linear')
      xw = window.*detrend(x(index));
   else
      xw = window.*detrend(x(index),'constant');
   end
   index = index + (nwind - noverlap);
   Xx = abs(fft(xw,nfft)).^2;
   Spec = Spec + Xx;
   Spec2 = Spec2 + abs(Xx).^2;
end
% Select first half
if ~any(any(imag(x)~=0)),   % if x is not complex
   if rem(nfft,2),    % nfft odd
      select = (1:(nfft+1)/2)';
   else
      select = (1:nfft/2+1)';
   end
   Spec = Spec(select);
   Spec2 = Spec2(select);
else
   select = (1:nfft)';
```

```
end
freq_vector = (select - 1)*Fs/nfft;
% find confidence interval if needed
if (nargout == 3)|((nargout == 0)&~isempty(p)),
    if isempty(p),
       p = .95;    % default
    end
    % Confidence interval from Kay, p. 76, eqn 4.16:
    % (first column is lower edge of conf int., 2nd col is upper edge)
    confid = Spec*chi2conf(p,k)/KMU;
    if noverlap > 0
       disp('Warning: confidence intervals inaccurate for NOVERLAP > 0.')
    end
end
Spec = Spec*(1/KMU);   % normalize
% set up output parameters
if (nargout == 3),
   Pxx = Spec;
   Pxxc = confid;
   f = freq_vector;
elseif (nargout == 2),
   Pxx = Spec;
   Pxxc = freq_vector;
elseif (nargout == 1),
   Pxx = Spec;
elseif (nargout == 0),
   if ~isempty(p),
      P = [Spec confid];
   else
      P = Spec;
   end
   newplot;
   plot(freq_vector,10*log10(abs(P))), grid on
   xlabel('Frequency'), ylabel('Power Spectrum Magnitude (dB)');
end
```

在上面的函数文件中，首先利用函数 nargin 检查输入参数的个数，并根据检查结果给函数的参数进行赋值，最后利用函数 nargout 判断返回参数的个数，来确定功率谱的输出形式。

1.4.3 echo、input、keyboard、pause 命令

通常情况下，执行 M 文件时，其命令不会在屏幕上显示，但利用 echo 命令可使命令在屏幕上显示，这给调试工作及演示带来了方便。利用 input 函数可在执行 M 文件过程中输入一行信息。例如，有一M 文件 pow2.m，其内容为：

```
n=input('please input the number of pow:');
2^n
```

当执行 pow2.m 时，得到提示信息并等待输入 n 值：

```
pow2
please input the number of pow: 4
ans
   16
```

input 语句还可用于输入表达式或字符串，并实现菜单的功能。

keyboard 语句与 input 语句类似，但其功能更强，它可等待输入多行命令，并将键盘输入的内容作为一个底稿文件来处理。keyboard 的这一特点非常有利于 M 文件的调试，另外也可用于在程序执行期间修改变量。例如建立一个函数文件 user.m：

```
function y=user(x)
%you can make this function
%to be any function of x
keyboard
```

当在命令窗口中执行这个函数时，可得到如下等待键盘输入的字符：

```
user
K>>
```

如果输入下面的语句：

```
K>>x=0:.01:1;
K>>plot(x,user(x))
K>>y=x.^3;
K>>return
```

就会得到如图 1.38 所示的图形，return 命令返回正常状态，表示文本输入结束，这一命令是必须的。

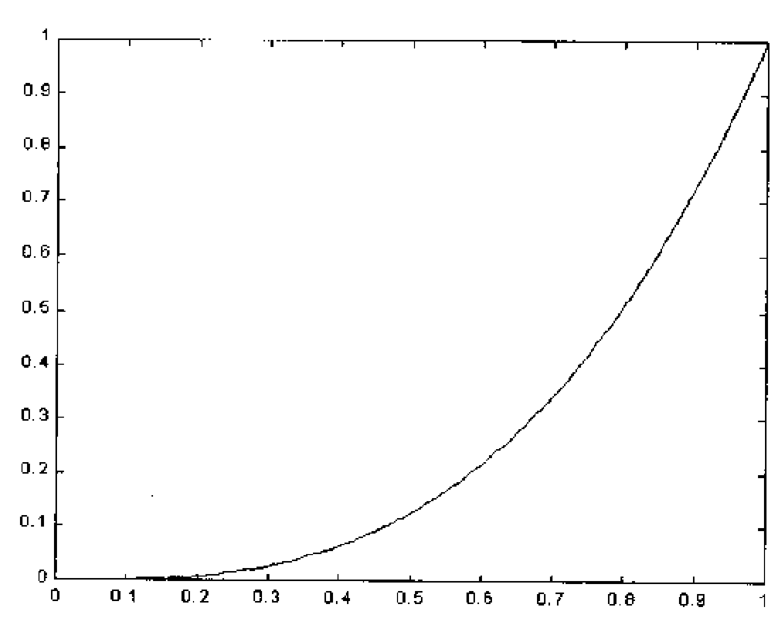


图1.38 keyboard命令得到的图形

如果想要画出另一种函数曲线，只需重复以上的步骤，只是输入 keyboard 的内容不

同而已。

pause 命令的功能是使程序暂停执行，当用户按任意键时，程序继续执行。pause(n)是等待 n 秒后自动继续执行。

1.4.4 提高速度及内存管理

MATLAB 中基于向量与矩阵的操作在速度上比基于编译器/解释器的操作要快一个数量级，这意味着要提高 MATLAB 的执行速度，必须将 M 文件中的算法尽量向量化。只要可能，尽量将 for 及 while 循环语句改成矢量或矩阵操作。例如，要计算 1000 点的余弦序列值，一种常见算法如下：

```
i=0;
for n=0:.01:9.99
i=i+1;
y(i)=cos(n);
end
```

完成上述相同的功能，可采用下面简单的向量化算法：

```
n=0:.01:9.99;
y=cos(n);
```

虽然两种算法实现的功能相同，但向量化算法要比常见算法快得多，因此，应尽量采用向量化算法。但在优化比较复杂的程序时，执行速度的提高并不总是明显的。有一点值得注意，如果速度是至关重要的，那么就应该尽可能地寻找向量化的方法。

函数 clear 可以清除基本工作空间中的变量，或清除指定的函数或变量。函数 save 可以有选择地保存内存空间中的变量，或保存基本工作空间中的全部变量。函数 quit 可以退出 MATLAB 运行，并释放所有占用的内存资源。load 函数可以将 M 文件调入内存，直接使用工作空间中的变量。在编写较大的程序时，使用这些函数的组合，可以对内存进行较好的管理。

另外有一个函数 pack，可用于整理内存空间。当在命令行中经常执行新建数组、改变数组大小等操作时，虽然系统动态分配内存的功能可以保证自由内存的数量，但有时在创建一个很大的数组时会出现内存不足的问题。出现内存不足的情况时，又不能删除工作空间中的变量，可以使用函数 pack 对内存空间进行一次整理。直接在命令行中使用函数 pack，相当于执行以下 4 个步骤：

(1) 将所有变量保存在磁盘上的临时文件 pack.tmp 中。

(2) 从内存中清除所有变量与函数。

(3) 用临时文件 pack.tmp 将保存的变量与函数重新载入内存。

(4) 删除临时文件 pack.tmp。

函数的嵌套调用与多次分行调用所用的内存相等。例如：

函数嵌套调用 result=function2(function1(input1))与下面两行占用的内存相同：

```
result=function1(input1);
result=function2(result);
```

在对一个变量赋值时，仅当这个变量在工作空间中不存在时，才对其分配存储空间，如果大小不同，则要改变其大小。例如：

```
x=10;%如果变量 x 不存在，则分配内存空间
x(8)=2;%如果 x 的第 8 个元素存在，则不分配内存空间
```

因此，应尽量使用已经创建的变量。

在同一个函数中同时使用一个变量作为输入和输出参数，这个变量在使用时将被复制。例如：

```
y=fun(x,y);%由于变量 y 同时作为输入或输出参数，因此在使用时被复制
```

Windows 用户除使用以上函数对 MATLAB 占用的内存进行管理以外，还要注意关闭一些不用的非活动窗口，以释放更多的内存空间。

1.5 MATLAB 6 的稀疏矩阵

稀疏矩阵是一种特殊类型的矩阵，即矩阵中包括较多的零元素。MATLAB 可以只保存矩阵中的非零元素及其在矩阵中的位置。在用稀疏矩阵进行计算时，通过消去零元素可以减少计算的时间。

1.5.1 稀疏矩阵的存储

MATLAB 对稀疏矩阵的存储有两种模式：完全存储与稀疏存储。

完全存储模式就是将矩阵的所有元素按其类型存储，因此一个 n×m 的实数矩阵，需要 n×m 个实数存储单元。

稀疏存储模式只保存矩阵中的非零元素及其相应的列号。存储时按列号存储，对每一列，非零元素存储在一列实数数组中(如为复数矩阵，则存储在两列实数数组中)，对应的行号、列号分别存储在两个整型数组中。例如对如下稀疏矩阵：

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5 \\ 4 & 0 & 2 & 0 \\ 0 & 4 & 6 & 0 \end{bmatrix}$$

可分别存储在 3 个数组(a,i,j)中：

```
a=[4 4 1 2 6 5]%非零元素
i=[3 4 1 3 4 2]%非零元素所在的行号
j=[1 2 3 3 3 4]%非零元素所在的列号
```

若用 8 位数来表示实数，整数用 4 位数来表示，则按完全存储方式，需要占用 160Byte，然而按稀疏存储只占用 96Byte，其稀疏密度为 6/20=30%。随着稀疏密度的降低，可进一步减少存储量。如一个 1000×1000 的实数矩阵，完全存储需占用 8×10^6 Byte。若稀疏密度为 1‰，则内存为 1.6×10^4Byte，只占用原来的 2‰。

1.5.2 创建稀疏矩阵

函数 full 与 sparse 是一对用来对矩阵存储模式进行转换的内部函数。对任一矩阵 x，full(x)将产生以完全方式存储的矩阵，当 x 原本就是完全存储方式时，则返回值就是 x 本身；当 x 是以稀疏存储时，则 full(x)返回值就是完全存储方式的矩阵。相反，sparse(x)会删去 x 的所有零元素，返回按稀疏模式存储的矩阵；如果 x 本身就是稀疏矩阵，则不变。

1. 直接创建稀疏矩阵

使用函数 sparse 可以用一组非零元素直接创建一个稀疏矩阵，其格式如下：

```
S = Sparse(i,j,s,m,n)
```

其中 i 与 j 都为数组，分别代表矩阵中非零元素的行号和列号；s 是一个全部元素为非零的数组，元素在矩阵中排列的位置为(i,j)；m 为输出的稀疏矩阵的行数，n 为输出的稀疏矩阵的列数。

例如，输入下面语句：

```
i=[3 4 1 3 4 2];
j=[1 2 3 3 4 4];
s=[1 3 4 6 2 5];
m=4;
n=4;
S=sparse(i,j,s,m,n)
```

得到如下结果：

```
S =
    (3,1)        1
    (4,2)        3
    (1,3)        4
    (3,3)        6
    (2,4)        5
    (4,4)        2
```

利用 full 函数可将上面的稀疏矩阵转换成一般矩阵：

```
A=full(S)
A =
     0     0     4     0
     0     0     0     5
     1     0     6     0
     0     3     0     2
```

函数 sparse 还有一种格式为：

```
S = Sparse(i,j,s,m,n,nzmax)
```

参数 i、j、s、m、n 的说明与上面的格式相同，参数 nzmax 用来设置矩阵中非零元素的最大数目，例如上面的例子若改如下：

```
i=[3 4 1 3 4 2];
j=[1 2 3 3 4 4];
s=[1 3 4 6 2 5];
m=4;
n=4;
nzmax=12;
S1=sparse(i,j,s,m,n,nzmax)
S2=full(S1)
S1 =
     (3,1)        1
     (4,2)        3
     (1,3)        4
     (3,3)        6
     (2,4)        5
     (4,4)        2
S2 =
     0     0     4     0
     0     0     0     5
     1     0     6     0
     0     3     0     2
```

比较以上两种格式的结果，发现稀疏矩阵 S 与 S1 的结果相同。在工作空间中查看两稀疏矩阵的结果，发现虽然它们的大小一样，但占用的字节数不同，如图 1.39 所示。

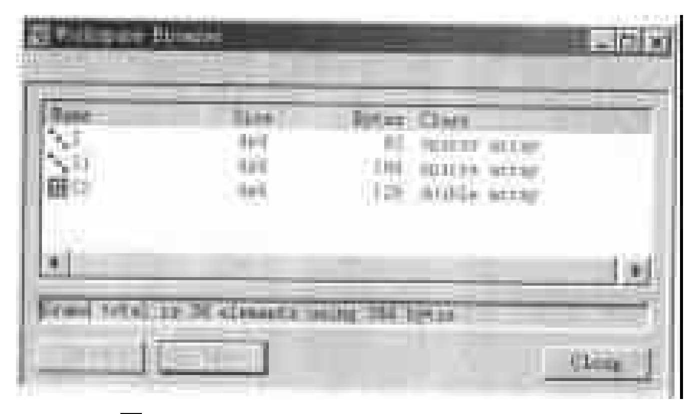


图1.39 两种创建稀疏矩阵格式的比较

2. 从对角线元素中创建稀疏矩阵

将一个矩阵的对角线元素保存在一个稀疏矩阵中，可以使用函数 spdiags 实现，其语法格式为：

```
S=spdiags(B,d,m,n)
```

创建一个大小为 m×n 的稀疏矩阵 S，其非零元素来自矩阵 B 中的元素且按对角线排列。

参数 d 指定矩阵 B 中用于生成稀疏矩阵 S 的对角线位置。矩阵的主对角线可以认为是第 0 条对角线，每向右上移动一条对角线编号加 1，向左下移动一条对角线编号减 1，也就是说 B 中 j 列的元素填充矢量 d 中第 j 个元素所指定的对角线。

例如，下面的语句由矩阵 B 与矢量 d 创建一个 7 行 3 列的稀疏矩阵 S：

```
B=[1 2 3;4 5 6;0 7 8];
d=[-1 0 1]';
S=spdiags(B,d,7,3)
```

得到的稀疏矩阵 S 为：

```
S =
   (1,1)        2
   (2,1)        1
   (1,2)        6
   (2,2)        5
   (3,2)        4
   (2,3)        8
   (3,3)        7
```

为更清楚地看出对角线位置的元素排列特点，利用函数 full 将稀疏矩阵 S 转换成一般形式的矩阵 S1：

```
S1-full(S)
S1 =
     2     6     0
     1     5     8
     0     4     7
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

用稀疏矩阵可以方便地表示二次操作，对角线上的元素为-2s，对角线下方或上方的元素为 1s，例如：

```
n=6;
A=sparse(1:n,1:n,-2*ones(1,n),n,n);
B=sparse(2:n,1:n-1,ones(1,n-1),n,n);
S=B+A+B'
S1=full(S)
S =
   (1,1)       -2
   (2,1)        1
   (1,2)        1
   (2,2)       -2
   (3,2)        1
   (2,3)        1
   (3,3)       -2
   (4,3)        1
   (3,4)        1
   (4,4)       -2
   (5,4)        1
   (4,5)        1
   (5,5)       -2
   (6,5)        1
   (5,6)        1
```

```
   (6,6)        -2
S1 =
    -2     1     0     0     0     0
     1    -2     1     0     0     0
     0     1    -2     1     0     0
     0     0     1    -2     1     0
     0     0     0     1    -2     1
     0     0     0     0     1    -2
```

3. 从外部文件中导入稀疏矩阵

用外部文件创建的文本文件，如果其中的数据按 3 个列排列，可以将这个文本文件载入工作空间，用于创建一个稀疏矩阵。

例如，有一文本文件 sparsem.dat，用 load 命令将其载入工作空间：

```
load d:\seaclutter\seadata\sparsem.dat
```

在当前空间中创建了一个变量 sparsem，用其第 1 列作为行号，第 2 列作为列号，第 3 列的非零元素作为稀疏矩阵的元素，创建一个稀疏矩阵 S：

```
dd =
     1     1     2
     2     1     0
     2     2     3
     2     3     4
     3     4     5
     3     3     0
     4     2     1
     5     3     2
     5     5     0
S=spconvert(dd)
S1=full(S)
S =
   (1,1)        2
   (2,2)        3
   (4,2)        1
   (2,3)        4
   (5,3)        2
   (3,4)        5
S1 =
     2     0     0     0     0
     0     3     4     0     0
     0     0     0     5     0
     0     1     0     0     0
     0     0     2     0     0
```

1.5.3 稀疏矩阵的操作

大多数 MATLAB 的标准数学函数都可以处理稀疏矩阵，此时可以将稀疏矩阵当作一

般矩阵看待。MATLAB 也提供了专门针对稀疏矩阵的函数。处理稀疏矩阵时，计算的复杂程度与稀疏矩阵中的非零元素的个数成正比，计算的复杂程度也与矩阵的行列大小有关，稀疏矩阵的乘法、乘方，包含一定次数的线性方程组等，都是比较复杂的运算。

1. 稀疏矩阵的交换

稀疏矩阵的行交换与列交换可以用以下两种方法表示：

- 对于交换矩阵 P，对稀疏矩阵 S 的行交换可表示为 P*S，列交换可表示为 S*P'。
- 对于一个交换矢量 p，p 为一般矢量，包含 1～n 个自然数的一个排列。对稀疏矩阵进行行交换，可以表示为 S(p,:)。S(:,p)为列交换形式。对于矩阵 S 的第 i 列进行行交换的形式为 S(p,i)。

例如，输入下面语句：

```
%创建稀疏矩阵
i=[3 4 1 3 4 2];
j=[1 2 3 3 4 4];
s=[1 3 4 6 2 5];
m=4;
n=4;
S=sparse(i,j,s,m,n);
%对稀疏矩阵进行行变换
p=[4 3 1 2];
A=S(p,:);
%对稀疏矩阵的第 2 列进行变换
v=S(p,2);
%转换成一般矩阵形式
S1=full(S)
A1=full(A)
v1=full(v)
```

得到如下结果：

```
S1 =
     0     0     4     0
     0     0     0     5
     1     0     6     0
     0     3     0     2
A1 =
     0     3     0     2
     1     0     6     0
     0     0     4     0
     0     0     0     5
v1 =
     3
     0
     0
     0
```

2. 稀疏矩阵的分解

与一般矩阵一样，对于稀疏矩阵同样可以进行 LU 分解、Cholesky 分解、QR 分解及一些不完全分解。

- LU 分解

如果 S 为稀疏矩阵，则下面的表达式可以产生 3 个矩阵 L、U 和 P，使 P×S=L×U 成立：

```
[L, U, P]=lu(S);
```

函数 lu 采用高斯消去法获得这 3 个矩阵，其中 P 为交换矩阵，它仅包含 n 个非零元素，矩阵 L 和 U 与一般密集矩阵的分解结果相同，L 为下三角矩阵，U 为上三角矩阵。稀疏矩阵的 LU 分解重点在于数值的稳定性，而不在于矩阵的稀疏性，上述 3 个返回的矩阵都为稀疏矩阵。

实际上稀疏矩阵与一般矩阵的 LU 结果相同，只不过在所需的时间与贮存空间上有很大的不同。另外，MATLAB 在进行分解时自动地为稀疏矩阵 L、U 分配内存，而不是在分解之前决定所需的内存并设置数据结构。

例如，输入下面的语句：

```
%创建稀疏矩阵
S=bucky;
        %进行 LU 分解
        [L,U]=lu(S);
        %利用 spy 函数显示分解结果
        figure(1)
spy(L)
        figure(2)
spy(U)
```

执行后得到如图 1.40 与图 1.41 所示的图形。

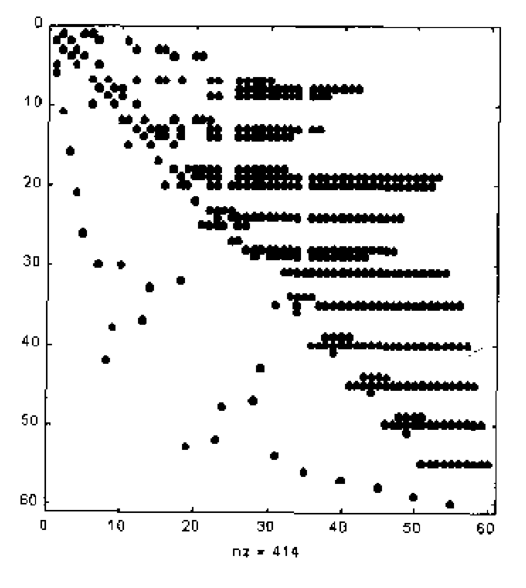


图1.40 LU分解后的上三角矩阵

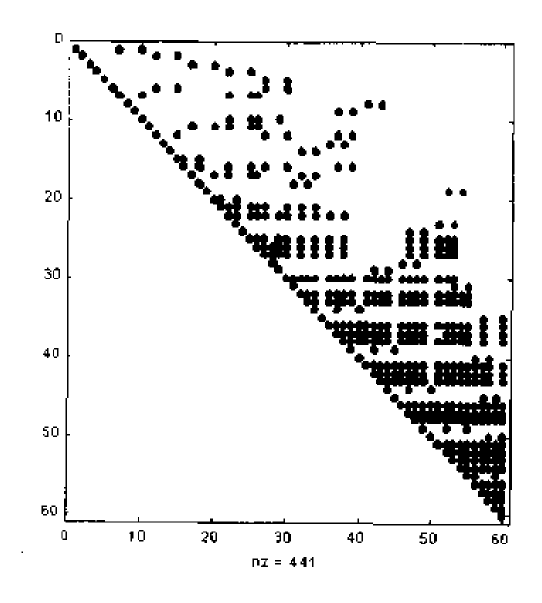


图1.41 LU分解后的下三角矩阵



- Cholesky 分解

如果矩阵 S 是一个对称的正定稀疏矩阵，则 Cholesky 分解如下：

```
R=chol(S)
```

可以返回一个上三角稀疏矩阵 R，使 R'×R=S。

例如，下面的语句对对称的正定稀疏矩阵 S 进行 Cholesky 分解：

```
S=[11 1 000; 1 22 100; 01 33 10; 00 1 44 1; 000155]
R=ch01(s)
```

得到如下结果：

```
S=11  1  0  0  0
   1  22  1  1  1
```

函数 chol 可对一般的正定对称矩阵进行分解，它不能自动地利用稀疏矩阵的优点，但可以先求出矩阵最小化带宽的排列矩阵，或最小化度的排列矩阵以提高分解的速度，并减小内存占用。Cholesky 算法不是主要针对稀疏矩阵和数据稳定性的，它是一个力求快速分解矩阵的算法，在开始分解时，就计算所需的内存并进行分配。

- QR 分解

MATLAB 可以对稀疏矩阵 S 进行完全 QR 分解，将矩阵分解为正交矩阵 Q 和三角形矩阵 R：

```
[Q,R]=qr(S);
```

但这样对稀疏矩阵进行分解一般是不现实的，因为正交矩阵中零元素的比例一般来说是不多的。因此通常使用所谓"不带 Q 的 QR 分解法"，仅有一个稀疏矩阵作为输入参数和一个输出参数，格式如下：

```
R=qr(S);
```

仅返回 QR 分解的上三角矩阵 R，使 R'×R=S'×S。

- 不完全分解

MATLAB 中提供了两种近似不完全分解函数：luinc 与 cholinc，它们也可用于稀疏矩阵的迭代方法。

函数 luinc 提供两种格式的不完全 LU 分解：

◆ LUINC(X,OPTS)

用于计算矩阵 X 的不完全 LU 分解，参数 OPTS 指定不完全 LU 分解的方式，共有 4 种方式可以选择：

(1) DROPTOL：为一非负的标量，用于指定不完全 LU 分解的容许误差。
(2) MILU：修正的不完全 LU 分解，其值要么为 1，代表采用修正的不完全 LU 分解，要么为 0，代表采用一般的不完全 LU 分解，默认值为 0。
(3) UDIAG：取值为 1 或 0。如果为 1，则表示用容许误差代替上三角矩阵对角线上的 0 值，以避免产生奇异因式。默认值为 0。
(4) THRESH：取值在[0，1]区间上，用于指定旋转门限。

例如下面的语句：

```
load west0479
A = west0479;
a=nnz(A)
b=nnz(lu(A))
c=nnz(luinc(A,1e-6))
```

得到如下结果:

```
a =
        1887
b =
       16777
c =
       10311
```

说明 A 有 1 887 个非零元素，A 的完全 LU 分解有 16 777 个非零元素，A 的容许误差小于 1e-6 时的不完全 LU 分解有 10 311 个非零元素。

◆ [L,U,P] = LUINC(X,'0')

用于计算矩阵 X 的不完全 LU 分解参数

U 是上三角矩阵，L 为一单位下三角矩阵的变换，P 是变换矩阵。例如:

```
load west0479
A = west0479;
[L,U,P] = luinc(A,'0');
isequal(spones(U),spones(triu(P*A)));
spones(L) ~= spones(tril(P*A));
D = (L*U) .* spones(P*A) - P*A
D =
  1.0e-014 *
  (12,7)       0.0014
  (346,7)      0.0014
  (347,7)      0.0014
  (82,10)     -0.0002
  (85,11)     -0.0056
  (52,47)     -0.0007
  (117,87)    -0.0003
  (135,88)     0.0056
  (205,93)     0.0111
  (361,93)     0.0111
 (407,93)     -0.0111
 (414,93)      0.0111
 (415,93)     -0.0111
 (418,93)     -0.0111
 (430,93)     -0.0111
 (477,93)     -0.0111
 (130,96)     -0.0014
 (394,96)     -0.0111
 (146,104)     0.0111
 (147,104)     0.0111
 (148,104)     0.0111
```

```
(353,104)    0.7105
(354,104)   -0.0111
(219,109)   -0.0111
(303,111)    0.0056
(322,116)   -0.0111
(123,120)    0.0111
(334,120)    0.0111
(249,122)    0.0000
(238,128)   -0.0007
(134,134)   -0.0111
(269,134)   -0.0002
(307,134)   -0.0002
(288,135)   -0.0111
(239,142)    0.0000
(411,166)    0.0000
(398,176)   -0.0003
(222,199)   -0.0056
(222,200)    0.0056
(408,201)    0.0056
(234,219)   -0.0016
(351,219)   -0.0003
(232,227)   -0.0007
(232,232)   -0.0666
(219,235)   -0.0111
(232,235)    0.0083
(234,235)   -0.0059
(235,235)   -0.0222
(253,235)   -0.0274
(251,251)   -0.0666
(253,254)   -0.0034
(410,254)    0.0145
(270,270)   -0.0333
(254,273)   -0.0111
(272,273)    0.0009
(273,273)   -0.0111
(410,273)    0.0102
(289,289)    0.0555
(272,292)    0.0044
(291,292)   -0.0015
(292,292)    0.0222
(308,292)   -0.0028
(311,303)   -0.0002
(308,308)    0.0777
(309,308)   -0.0111
(292,311)   -0.0111
(310,311)    0.0021
(311,311)    0.0056
(117,322)    0.0083
(303,322)   -0.0028
(311,322)    0.0016
(322,322)   -0.0024
```

```
(434,432)     -0.0111
(472,432)     -0.0111
```

3. 稀疏矩阵的特征值与奇异值

与一般矩阵的特征值求解函数 eig 不同的是，计算稀疏矩阵的特征值采用函数 eigs。一般矩阵的奇异值分解用函数 svd，对稀疏矩阵的奇异值分解使用函数 svds。

- 计算稀疏矩阵的特征值

  计算稀疏矩阵特征值的函数 eigs 共有下面 4 种格式：

  - [V,D,FLAG] = EIGS(A)

    返回矩阵的部分特征值与特征向量，A 是方阵，可以是稀疏矩阵或一般矩阵，对称矩阵或不对称矩阵，实矩阵或复矩阵。

  - [V,D,FLAG] = EIGS('Afun',N)

    A 是包含 M 文件的字符串，N 指定特征值问题的次数。

  - [V,D,FLAG] = EIGS(A,B,K,SIGMA)

    参数 A 与第一种格式相同。B 为与 A 大小相同的对称正定矩阵，如果参数 B 是默认矩阵，则 B=eye(size(A))，如果指定 B，则采用 Cholesky 分解。K 为特征值的个数。参数 SIGMA 如果不指定，则计算 K 个最大特征值，如果 SIGMA 为 0，则计算 K 个最小特征值，另外，SIGMA 还可为下述字符之一：

    'LM'：等效于默认状态；

    'SM'：等效于 SIGMA 为 0；

    'LR'：计算最大实部；

    'SR'：计算最小实部；

    'BE'：从谱的两端计算 K/2 个特征值。

  - [V,D,FLAG] = EIGS('Afun',N,B,K,SIGMA)

    参数'Afun'与 N 的说明见格式 2，参数 B 与 SIGMA 的说明见格式 3，参数 K 若不指定，则计算 K = MIN(N,6)个特征值。

例如，计算矩阵 west0479 的 6 个最大特征值及其对应的特征向量：

```
load west0479;
[vlm dlm]=eigs(west0479,6,'sm');
```

工作空间浏览器中显示的结果如图 1.42 所示。

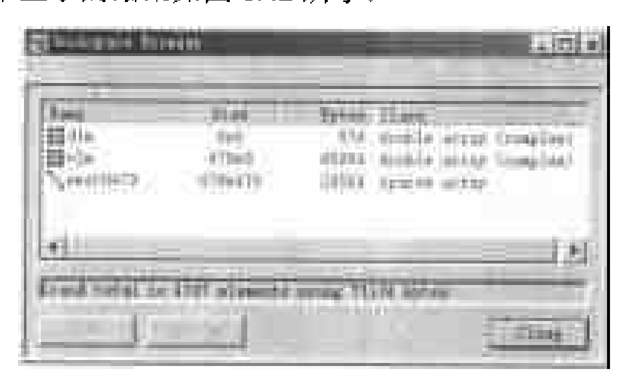


图1.42 工作空间浏览器中显示的结果

- 计算稀疏矩阵的奇异值

  计算稀疏矩阵奇异值的函数 svds 共有下面 3 种格式：

  - 1S = SVDS(A)，返回矩阵 A 的 6 个最大奇异值。
  - S = SVDS(A,K)，计算矩阵 A 的 K 个最大奇异值。等效于格式 S=SVDS(A,K, 'L')。
  - S = SVDS(A,K,SIGMA)，计算矩阵 A 离标量移位 SIGMA 最近的 K 个奇异值，例如 S = SVDS(A,K,0)计算 K 个最小奇异值。

例如，计算 479 阶稀疏方阵的奇异值：

```
load west0479;
s1 = svds(west0479,10)
ss = svds(west0479,10,0)
s2 = svds(west0479,10,2)
```

得到如下结果：

```
s1 =
  1.0e+005 *
    3.1895
    3.1725
    3.1695
    3.1685
    3.1669
    0.3038
    0.1467
    0.0528
    0.0458
    0.0424
ss =
  1.0e-003 *
    0.1379
    0.0999
    0.0918
    0.0712
    0.0562
    0.0517
    0.0450
    0.0402
    0.0042
    0.0010
s2 =
    2.0817
    2.0783
    2.0730
    2.0526
    2.0192
    2.0067
    1.9975
    1.9741
    1.9599
    1.9323
```

第 2 章　离散信号及其 MATLAB 实现

2.1　典型离散信号的表示方法

通过 MATLAB 内部一些简单函数如 zeros、ones 等及其基本操作，很容易产生常用的一些典型离散信号，需要注意的一点是，MATLAB 中只能产生有限长的序列，且其下标的约定从 1 开始递增。

1. 单位抽样序列

$$\delta(n)=\begin{cases}1 & n=0\\0 & n\neq 0\end{cases}$$

又称 Kronecker 函数，该信号在离散信号与离散系统的分析与综合中有着重要的作用。在 MATLAB 中可利用 zeros 函数来实现，如要产生 N 点的单位抽样序列，可通过以下语句实现：

$$x=zeros(1,N);$$
$$x(1)=1;$$

若将 $\delta(n)$ 在时间轴上延迟了 k 个抽样周期，得 $\delta(n-k)$，即

$$\delta(n-k)=\begin{cases}1 & n=k\\0 & n\neq k\end{cases}$$

如要产生 N 点序列：

$$\delta(n-k)=\begin{cases}0 & 1\leqslant n\leqslant k\\1 & n=k\\0 & k<n\leqslant N\end{cases}$$

则可采用如下 MATLAB 语句实现：

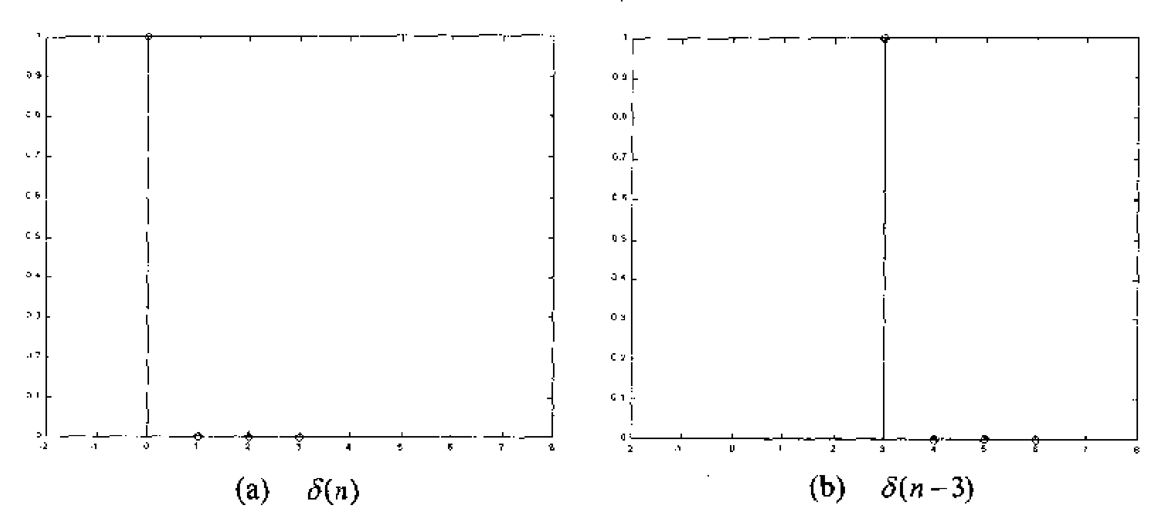
$$x=zeros(1,N);$$
$$x(k)=1;$$

一个 4 点的单位抽样信号及其右移 3 位所得序列如图 2.1 所示。

2. 单位阶跃序列

$$u(n)=\begin{cases}1 & n\geqslant 0\\0 & n<0\end{cases}$$





(a)　$\delta(n)$　　　　(b)　$\delta(n-3)$

图2.1　单位抽样信号及其移位

MATLAB 中的 ones 函数可以容易实现 N 点单位阶跃序列：

$$x = ones(1,N);$$

3.　正弦序列

$$x(n) = A\sin(2\pi f n T_s + \varphi)$$

采用 MATLAB 实现：

$$n = 0:N-1;$$
$$x = A * \sin(2 * pi * f * n * Ts + fai);$$

4.　复正弦序列

$$x(n) = e^{jwn}$$

采用 MATLAB 实现：

$$n = 0:N-1;$$
$$x = \exp(j * w * n);$$

5.　指数序列

$$x(n) = a^n$$

采用 MATLAB 实现：

$$n = 1:N;$$
$$x = a.\wedge n;$$

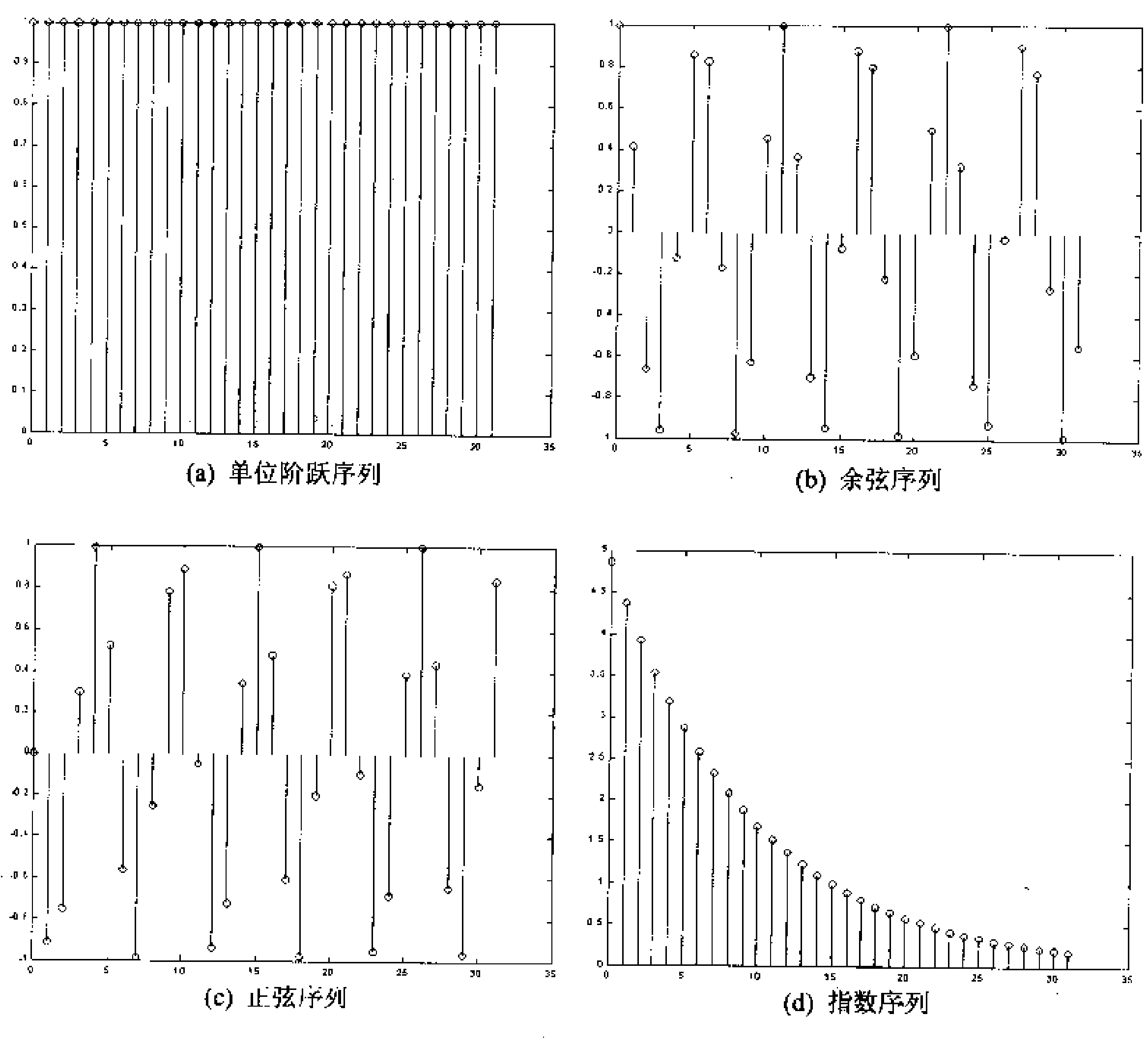
N 为 32 点的单位阶跃序列、余弦序列、正弦序列及指数序列，如图 2.2 所示。

6.　随机序列

MATLAB 中可以很容易产生如下两类随机信号：

- rand(1, N)产生[0，1]上均匀分布的随机序列；
- randn(1, N)产生均值为 0，方差为 1 的高斯随机序列，也就是白噪声序列。其他分布的随机数可通过上述随机数的变换而产生。图 2.3 为 MATLAB 函数产生的均匀

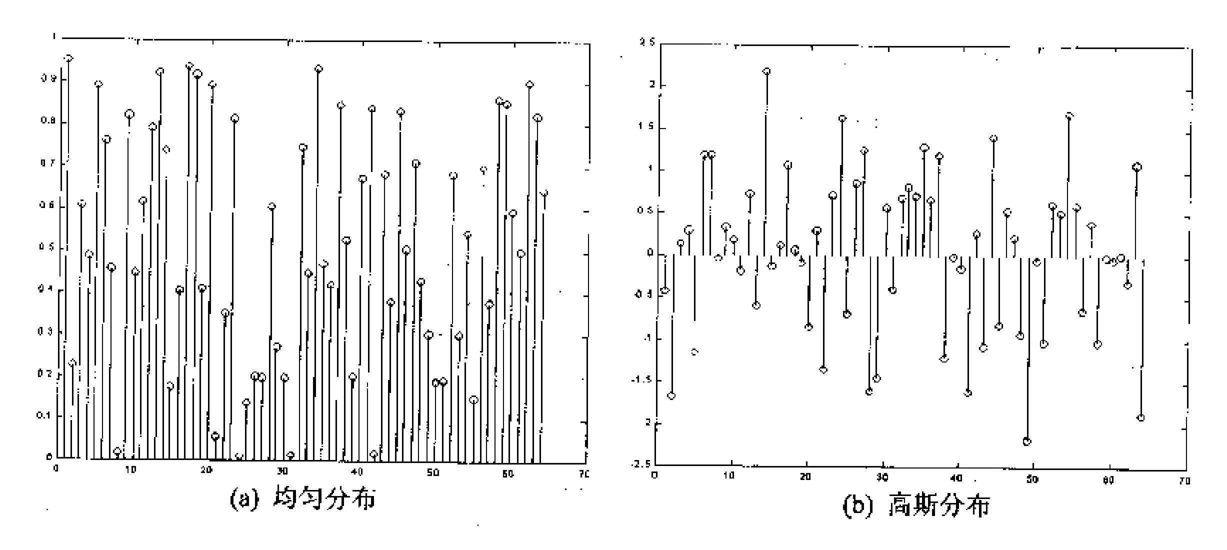
分布的随机序列与高斯随机序列。



(a) 单位阶跃序列　(b) 余弦序列

(c) 正弦序列　(d) 指数序列

图2.2　几个典型的离散序列



(a) 均匀分布　(b) 高斯分布

图2.3　随机序列

2.2　离散信号的基本运算

在数字信号处理中，对信号所作的基本运算分别是移位、相加、相乘等，下面分别通过实例利用 MATLAB 来实现。

- 信号的延迟

给定离散信号 $x(n)$，若信号 $y(n)$ 定义为：

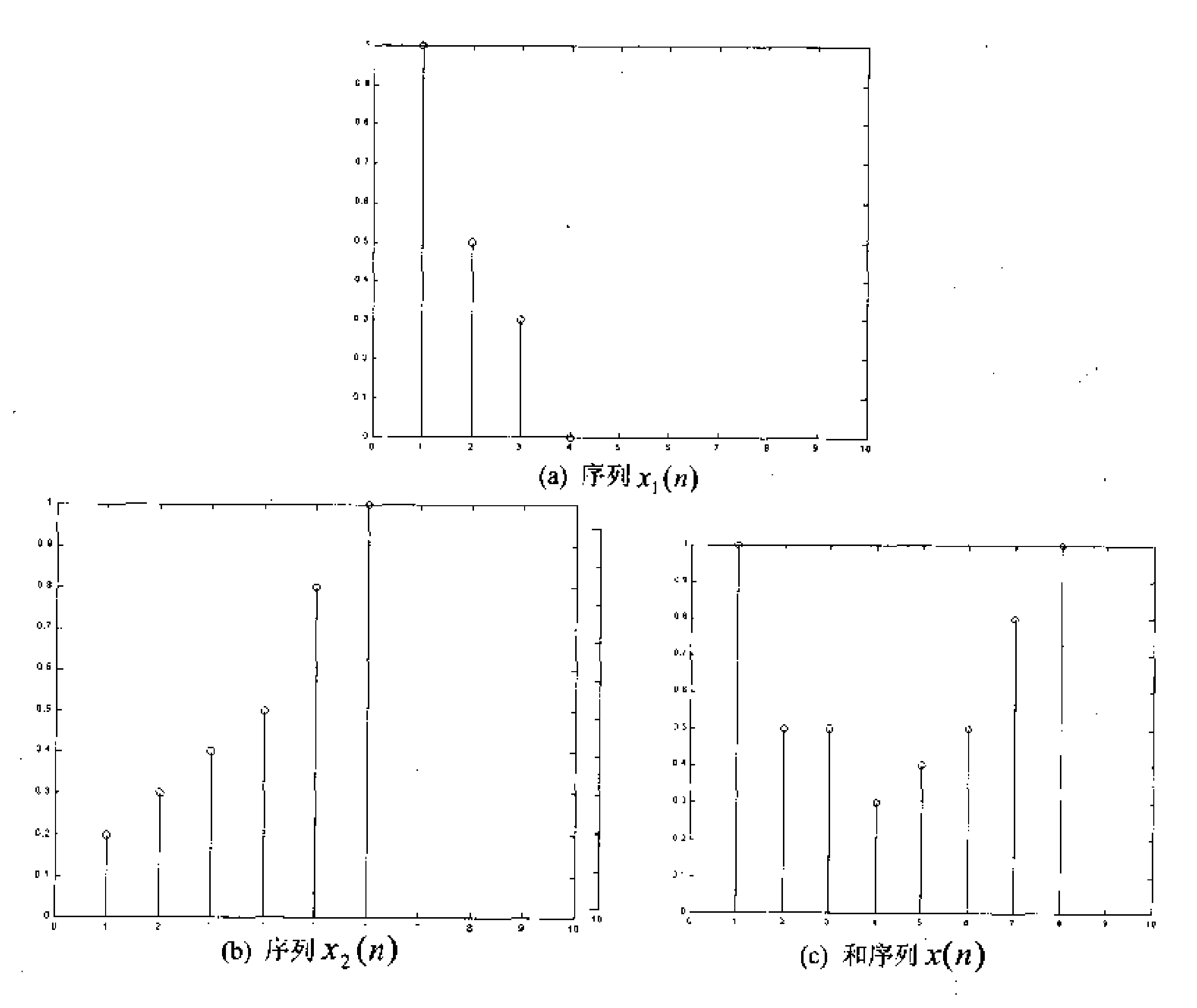
$$y(n) = x(n-k)$$

那么，$y(n)$ 是信号 $x(n)$ 在时间轴上右移 k 个抽样周期得到的新序列。

- 信号相加

$$x(n) = x_1(n) + x_2(n)$$

值得注意的是，当序列 $x_1(n)$ 和 $x_2(n)$ 的长度不等或位置不对应时，首先应使两者位置对齐，然后通过 zeros 函数左右补零使其长度相等后再相加。如图 2.4 所示的两序列 $x_1(n)$ 和 $x_2(n)$，相加后得到序列 $x(n)$。采用 MATLAB 实现如下：



(a) 序列 $x_1(n)$

(b) 序列 $x_2(n)$

(c) 和序列 $x(n)$

图2.4 信号相加

```
n1=1:4;
x1=[1 0.5 0.3 0];
```

```
n2=3:8;
x2=[0.2 0.3 0.4 0.5 0.8 1];
n=1:8;
x1=[x1 zeros(1,8-length(n1))];
x2=[zeros(1,8-length(n2)) x2];
x=x1+x2;
```

- 信号相乘

$$x(n)=x_1(n)x_2(n)$$

这是样本与样本之间的点乘运算，在 MATLAB 中可采用.*来实现，但两序列应做如相加运算同样的操作。序列 $x_1(n)$ 和 $x_2(n)$ 同上，相乘后得到序列 $x(n)$ 。MATLAB 实现时只需将程序的最后一行改为相乘运算即可。即：

```
n1=1:4;
x1=[1 0.5 0.3 0];
n2=3:8;
x2=[0.2 0.3 0.4 0.5 0.8 1];
n=1:8;
x1=[x1 zeros(1,8-length(n1))];
x2=[zeros(1,8-length(n2)) x2];
x=x1.*x2
```

得到的积序列为：

```
x =
   0    0    0.0600    0    0    0    0    0
```

- 信号的标量乘

$$y(n)=cx(n)$$

采用 MATLAB 很容易实现：

```
y=c*x;
```

例如，在 MATLAB 命令窗口中输入命令：

```
x=linspace(0,1,8)
c=4;

y=c*x
```

得到的序列及其标量积为：

```
x =
    0       0.1429    0.2857    0.4286    0.5714    0.7143
    0.8571  1.0000
y =
    0       0.5714    1.1429    1.7143    2.2857    2.8571
    3.4286  4.0000
```

- 信号的翻转

$$y(n)=x(-n)$$

在 MATLAB 中可直接用 fliplr 函数实现此操作。如将序列 x=[1 2 3]翻转得到 y=[3 2 1]，

只需一条语句：

```
y=fliplr(x);
```

即可实现。

● 信号和

对于 N 点信号 $x(n)$，其和的定义为：

$$y=\sum_{n=1}^{N}x(n)$$

采用 MATLAB 实现：

```
y=sum(x);
```

● 信号积

对于 N 点信号 $x(n)$，其积的定义为：

$$y=\prod_{n=1}^{N}x(n)$$

采用 MATLAB 实现：

```
y=prod(x);
```

● 信号的能量

有限长信号的能量定义为

$$E_X=\sum_{n=1}^{N}x(n)x^*(n)=\sum_{n=1}^{N}|x(n)|^2$$

在 MATLAB 中可有两种方法实现：

```
Ex=sum(x.*conj(x));
```

或

```
Ex=sum(abs(x).^2);
```

2.3 噪声及信号波形的产生

MATLAB 内部提供了大量的函数，用来产生噪声及常用的信号波形。本节首先介绍了噪声的产生方法，然后对数字信号处理中的常用信号如三角波、方波、线性调频信号的波形产生方法进行了阐述。

如 2.1 节所述，对于[0，1]上均匀分布的随机噪声可以直接利用 MATLAB 中的 rand 函数实现，均值为 0，方差为 1 的高斯随机噪声即白噪声由函数 randn 产生。对于其他分布(如瑞利分布、对数正态分布等)的随机噪声可以通过上述随机数的变换而产生。

MATLAB 中含有丰富的函数用以产生无线电技术及通讯等领域广泛采用的信号波形，如方波、三角波和线性调频信号等。下面结合具体示例说明函数的用法。

1. SAWTOOTH函数

该函数有 SAWTOOTH(T) 和 SAWTOOTH(T,WIDTH)两种格式，分别用来产生锯齿波和三角波。SAWTOOTH(T)对时间变量 T 产生周期为 2π 的锯齿波，类似于 SIN(T)，但其波形为锯齿状，幅度在+1～－1 之间变化。SAWTOOTH(T,WIDTH) 对时间变量 T 产生三角波，WIDTH 是在 0～1 之间取值的尺度参数，函数值在[0 WIDTH*2*pi]区间内由－1 增大到+1，在 [WIDTH*2*pi 1]区间内由+1 减小到－1。当 WIDTH=0.5 时，产生对称的三角波，当 WIDTH=1 时，就产生锯齿波，即 SAWTOOTH(T,1)相当于 SAWTOOTH(T)。

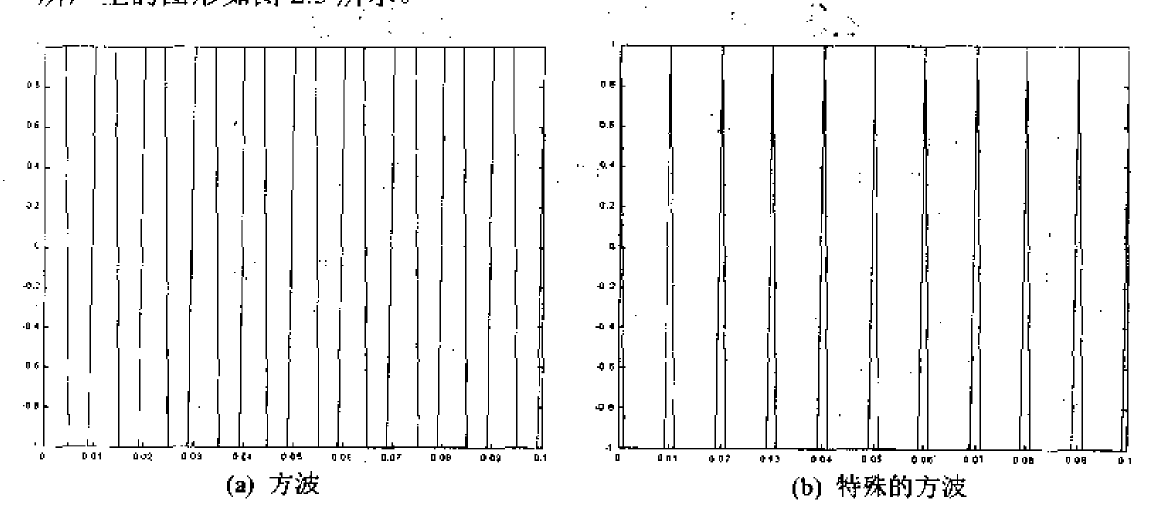
2. SQUARE函数

SQUARE 函数有 SQUARE(T)和 SQUARE(T,DUTY)两种格式，SQUARE(T)相对于时间变量 T 产生周期为 2π，幅值为±1 的方波。SQUARE(T,DUTY)产生指定周期的方波，DUTY 是信号为正值的区域在一个周期内所占的比例。

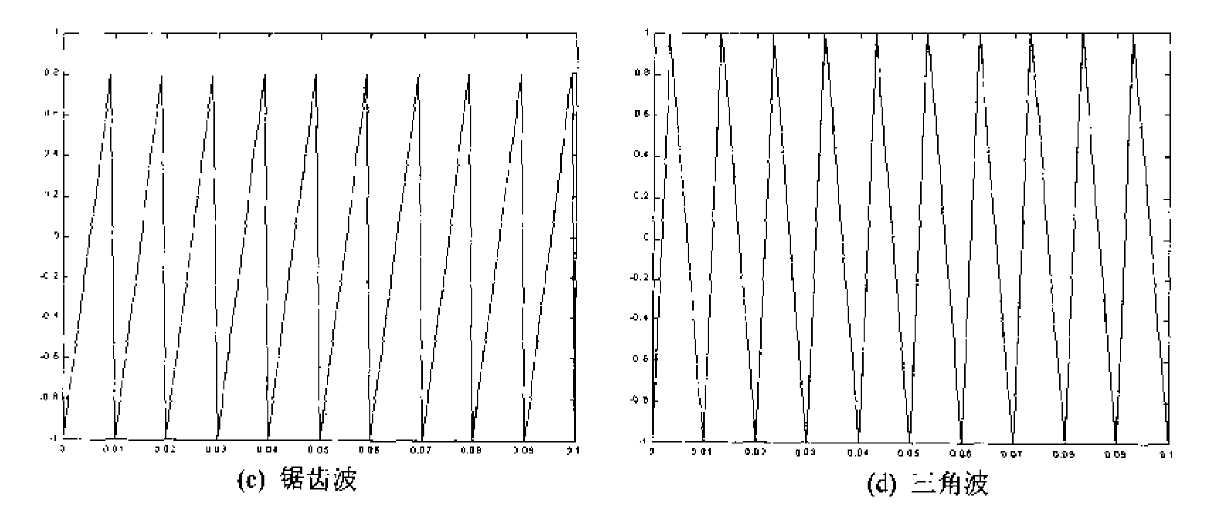
下面的语句产生频率为 100Hz 的方波、锯齿波与三角波。

```
f=100;%signal frequency
Fs=1000;%sample frequency
width=0.3;%scalar parameter of function SAWTOOTH
duty=0.5;%parameter of function SQUARE
t=0:1/Fs:0.1;
c=2*pi*f*t;
x=square(c);%generate square wave
x1=square(c,duty);%generate special square wave
y=sawtooth(c);%generate sawtooth wave
y1=sawtooth(c,width);generate triangle wave
%plot the figures
subplot(221) plot(t,x)
subplot(222) plot(t,x1)
subplot(223) plot(t,y)
subplot(224) plot(t,y1)
```

所产生的图形如图 2.5 所示。



(a) 方波　　(b) 特殊的方波



(c) 锯齿波　　(d) 三角波

图2.5 SQUARE函数和SAWTOOTH函数

3. SINC函数

SINC 函数的格式为 SINC(T)，对输入的变量 T 计算数学 sinc 函数的值，即

$$\sin c(t)=\begin{cases}1 & t=0\\ \dfrac{\sin(\pi t)}{\pi t} & t\neq 0\end{cases}$$

它是宽度为 2π，幅度为 1 的矩形脉冲的逆傅立叶变换：

$$\sin c(t)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\mathrm{e}^{j\omega t}dt$$

4. DIRIC函数

DIRIC 用来求取 Dirichlet 函数。Dirichlet 函数的定义如下：

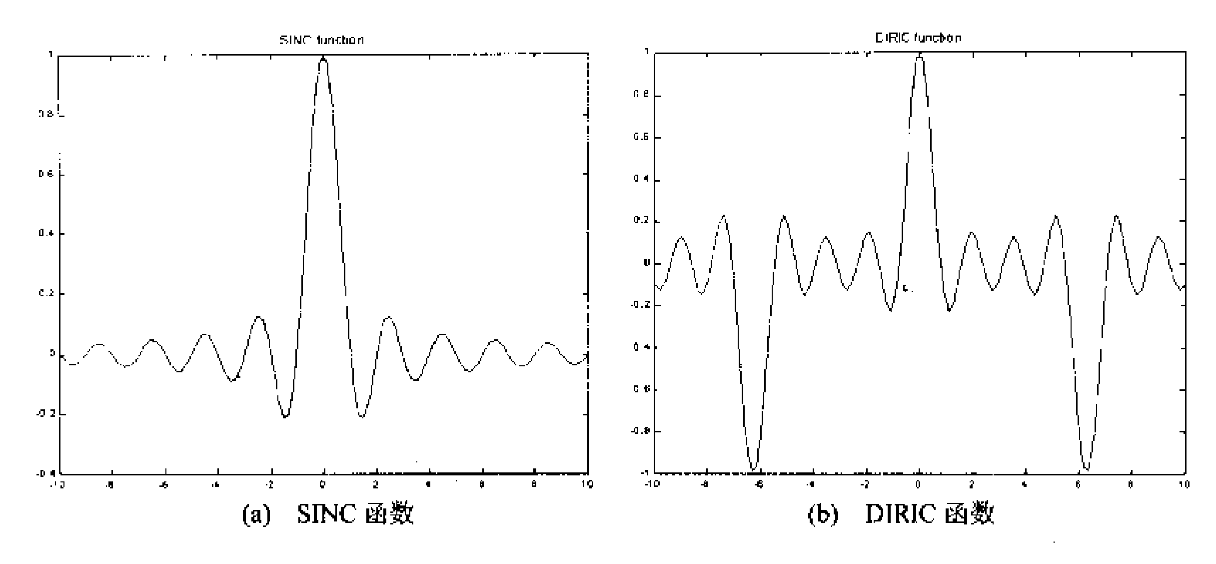
$$dirichlet(t)=\begin{cases}(-1)^{k(n-1)} & t=2\pi k,k=0,\pm1,\cdots\\ \dfrac{\sin(nt/2)}{n\sin(t/2)} & \text{其他}\end{cases}$$

其中 n 为正整数。当 n 为偶数时，Dirichlet 函数的周期为 4π，当 n 为奇数时，Dirichlet 函数的周期为 2π。

下面这段程序可分别得到 SINC 函数和 DIRIC 函数的曲线，如图 2.6 所示，从中可以看出两函数的区别。

```
t=linspace(-10,10);
x=sinc(t);
subplot(211)
plot(t,x)
title('SINC function');
subplot(212)
y=diric(t,8);
```

```
plot(t,y)
title('DIRIC function');
```



(a) SINC 函数　　(b) DIRIC 函数

图2.6 SINC函数和DIRIC函数

5. CHIRP函数

该函数用于产生调频的余弦脉冲。其用法如下：

- CHIRP(T,F0,T1,F1)

在时间范围 T 内产生线性调频脉冲，在 0 时刻信号的瞬时频率为 F0 赫兹，在 T1 时刻信号的瞬时频率为 F1 赫兹。

- CHIRP(T,F0,T1,F1,'METHOD')

METHOD 用来指定调频方法，可以提供的调频方法有线性、二次型和对数 3 种。其中默认方法为线性调频，值得注意的是，当采用对数调频时，必须满足 F1>F0。

- CHIRP(T,F0,T1,F1,'METHOD',PHI)

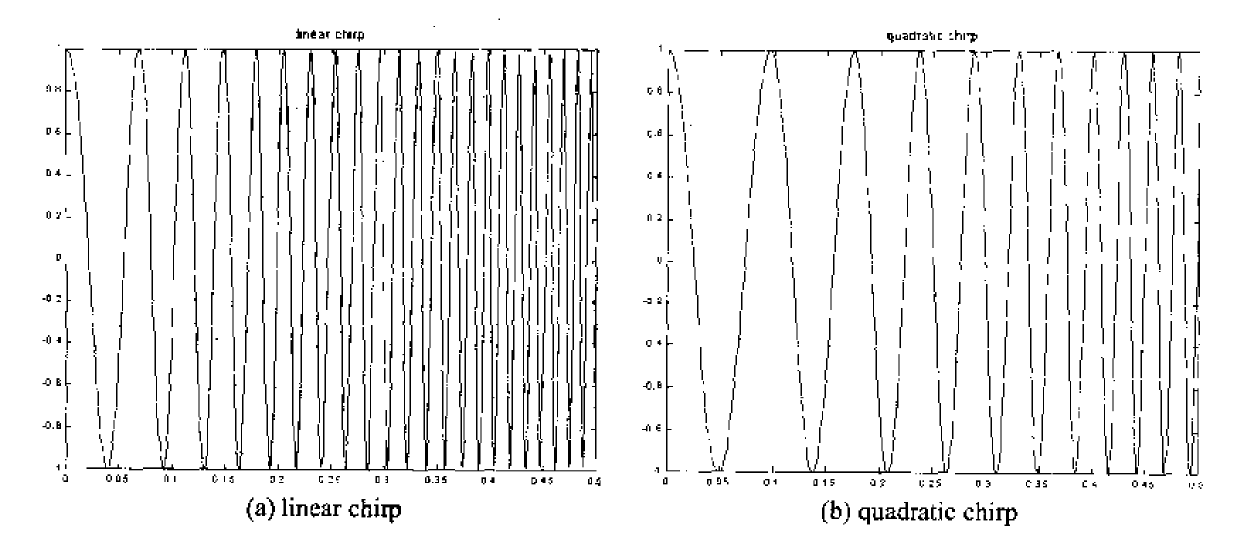
PHI 用来指定初始相位。

下面的语句可以产生线性调频与二次型调频信号。

```
Fs=1000;%sample frequency unit in Hz
t=0:0.001:2;
F0=10;%10Hz at t=0
T1=1;
F1=150;%cross 150Hz at t=1sec
y=chirp(t,F0,T1,F1);%generate a linear chirp
y1=chirp(t,F0,T1,F1,'q');%generate a quadratic chirp
figure(1)
plot(t,y)
title('linear chirp')
axis([0 0.5 -1 1])
figure(2)
plot(t,y1)
```

```
title('quadratic chirp')
axis([0 0.5 -1 1])
```

所得图形如图 2.7 所示。



(a) linear chirp　　(b) quadratic chirp

图2.7　CHIRP函数

第3章　离散系统及其MATLAB实现

在数字信号处理中，通常研究的离散系统大都为 LSI 系统。本章首先给出了离散系统的基本概念，然后对离散系统的时域与频域表示方法及其相应的 MATLAB 实现进行了详细的介绍，最后阐述了有关离散系统变换的知识。

3.1　离散系统的基本概念

一个离散系统，可以抽象为一种变换，或是一种映射，即把输入序列 $x(n)$ 变换为输出序列 $y(n)$：

$$y(n)=\mathrm{T}[x(n)] \tag{3.1.1}$$

式中 T 代表变换。这样，一个离散系统，既可以是一个硬件装置，也可以是一个数学表达式。总之，一个离散系统的输入、输出关系可用图 3.1 表示。

x(n) → T[ ] → y(n)

图3.1　离散系统

下面是有关离散系统的几个重要定义。

1.　线性

设一个离散系统对 $x_1(n)$ 的响应是 $y_1(n)$，对 $x_2(n)$ 的响应是 $y_2(n)$，即

$$\begin{aligned}y_1(n)&=\mathrm{T}[x_1(n)]\\ y_2(n)&=\mathrm{T}[x_2(n)]\end{aligned} \tag{3.1.2}$$

若该系统对 $\alpha x_1(n)+\beta x_2(n)$ 的响应是 $\alpha y_1(n)+\beta y_2(n)$，即

$$\begin{aligned}y(n)=\mathrm{T}[\alpha x_1(n)+\beta x_2(n)]&=\alpha\mathrm{T}[x_1(n)]+\beta\mathrm{T}[x_2(n)]\\ &=\alpha y_1(n)+\beta y_2(n)\end{aligned} \tag{3.1.3}$$

那么，称该系统是线性的。式中 α,β 是任意常数。显然，“线性”的含义是指该系统输入、输出之间满足叠加原理，如图 3.2 所示。

2.　移不变性

设一个离散系统对 $x(n)$ 的响应是 $y(n)$，即

$$y(n)=\mathrm{T}[x(n)]$$

若满足

$$\mathrm{T}[x(n-k)]=y(n-k) \tag{3.1.4}$$

则该系统是移不变的。同时具有线性和移不变的离散系统成为线性移不变系统，简称LSI系统。

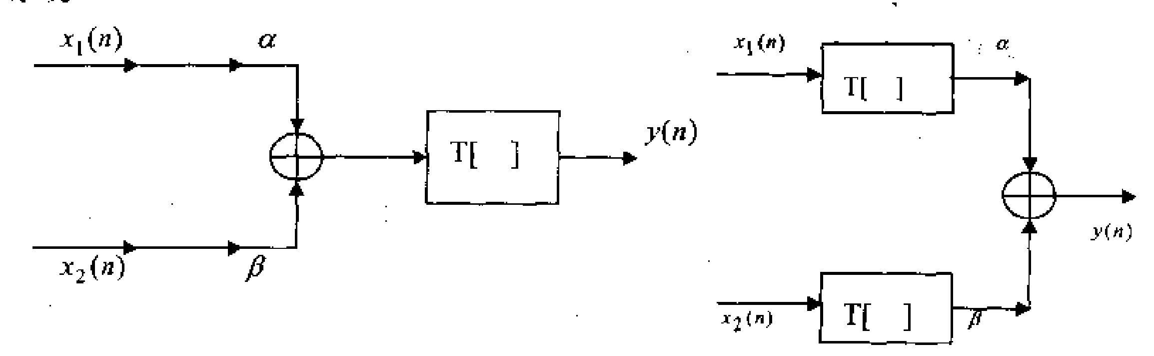


图3.2 线性系统定义的图解说明

3. 因果性

一个LSI系统，如果它在任意时刻的输出只决定于现在时刻和过去的输入，而与将来的输入无关，那么，该系统是因果系统。

4. 稳定性

一个信号$x(n)$，如果存在一个实数R，使得对所有的n都满足$|x(n)|\leqslant R$，那么，称$x(n)$是有界的。对一个LSI系统，若输入$x(n)$是有界的，输出$y(n)$也有界，那么该系统是稳定的。稳定性是一个系统能否正常工作的先决条件。

3.2 离散系统的表示方法

对于同一个离散系统，可以从时域和频域两个方面来进行描述，也可以对其进行内部描述。另外，频域描述又可分成频率响应、转移函数、零极点增益与二次分式几种不同的表示形式。

在数字信号处理中，通常研究的离散系统大都为LSI系统。

3.2.1 LSI系统的时域表示

一个LSI系统可用一个常系数线性差分方程来描述：

$$y(n)=-\sum_{k=1}^{N}a(k)y(n-k)+\sum_{r=0}^{M}b(r)x(n-r) \tag{3.2.1}$$

式中$a(k),k=1,\cdots,N,b(r),r=0,\cdots,M$是方程的系数。给定输入信号$x(n)$及系统的初始条件，可求出该差分方程的解$y(n)$，从而得到系统的输出。

若令输入信号$x(n)=\delta(n)$，这时的输出$y(n)$是由单位抽样信号$\delta(n)$激励该系统所产生

的响应，称为单位抽样响应 $h(n)$。$h(n)$ 反映了系统的固有特征，它是离散系统的一个重要参数。LSI 系统的输入输出关系可通过单位抽样响应 $h(n)$ 表示：

$$y(n)=x(n)*h(n)=\sum_{k=-\infty}^{\infty}x(k)h(n-k) \tag{3.2.2}$$

3.2.2 LSI 系统的频域表示

1. 频率响应

任意 LSI 系统都可由单位抽样响应 $h(n)$ 表示，相应地在频域中可用频率响应 $H(e^{jw})$ 来表示，它是 $h(n)$ 的离散傅立叶变换，即：

$$H(e^{jw})=\sum_{n=-\infty}^{\infty}h(n)e^{-jwn} \tag{3.2.3}$$

这样在频域输入输出关系可简单表示成：

$$Y(e^{jw})=X(e^{jw})H(e^{jw}) \tag{3.2.4}$$

由差分方程表示的 LSI 系统：

$$y(n)=-\sum_{k=1}^{N}a(k)y(n-k)+\sum_{r=0}^{M}b(r)x(n-r) \tag{3.2.5}$$

可通过输入 $x(n)=e^{jwn}$ 求出其频域函数：

$$H(e^{jw})=\frac{\sum_{r=0}^{M}b(r)e^{-jwr}}{1+\sum_{k=1}^{N}a(k)e^{-jwk}} \tag{3.2.6}$$

由于 $e^{jw}=e^{j(w+2k\pi)}$，所以 $H(e^{jw})$ 是周期为 2π 的周期函数。如果在[0 π]等间隔分成 $l=0,1,\cdots,L$，则有：

$$H(e^{jw_l})=\frac{\sum_{r=0}^{M}b(r)e^{-jw_lr}}{1+\sum_{k=1}^{N}a(k)e^{-jw_lk}} \tag{3.2.7}$$

2. 转移函数

若定义 $z=e^{jw}$，则 LSI 系统的频率响应变成：

$$H(z)=\sum_{n=-\infty}^{\infty}h(n)z^{-n} \tag{3.2.8}$$

该式为单位抽样响应 $h(n)$ 的 Z 变换，$H(z)$ 是系统的转移函数。

同样，由差分方程表示的 LSI 系统对应的转移函数为：

$$H(z)=\frac{\sum_{r=0}^{M}b(r)z^{-r}}{1+\sum_{k=1}^{N}a(k)z^{-k}} \tag{3.2.9}$$

3. 零极点增益

将转移函数的分子、分母分别作因式分解，便可得出 LSI 系统的零极点增益表示形式：

$$H(z)=g\frac{\prod_{r=1}^{M}(z-z_r)}{\prod_{k=1}^{N}(z-p_k)} \tag{3.2.10}$$

式中 g 称为系统的增益因子。使分母多项式等于零的 z 值，(即 P_k，k=1，2，3…，N)，称为系统的极点，同理，使分子多项式等于零的 z 值，(即 2r，r=1，2，…，M)，称为系统的极点。

通过系统的零极点增益表示形式，很容易判断一个 LSI 系统的稳定性，也就是说，若所有的极点都位于单位圆内，则系统是稳定的。同样，由零极图可以大致估计出系统的频率响应，通过零、极点分析还可得出滤波器设计的一般原则，详细内容可参看有关 DSP 的教材。

4. 二次分式

系统的零极点增益是将转移函数 $H(z)$ 的分子分母多项式分成了一阶多项式的连乘。考虑到 $H(z)$ 若有复数零、极点，那么，它们必然是共轭成对出现的。作物理实现时，其系数应为实数，因此，将它们分解成二次分式的形式更为合理。若 $N \geqslant M$，N 为偶数，则可将 $H(z)$ 分成 $L=N/2$ 个二次分式：

$$H_i(z)=\frac{1+b_{i1}z^{-1}+b_{i2}z^{-2}}{1+a_{i1}z^{-1}+a_{i2}z^{-2}}，\quad i=1,2,\cdots,L \tag{3.2.11}$$

的连乘：

$$H(z)=\prod_{i=1}^{L}H_i(z) \tag{3.2.12}$$

$H_i(z)$ 的信号流图如图 3.3 所示。

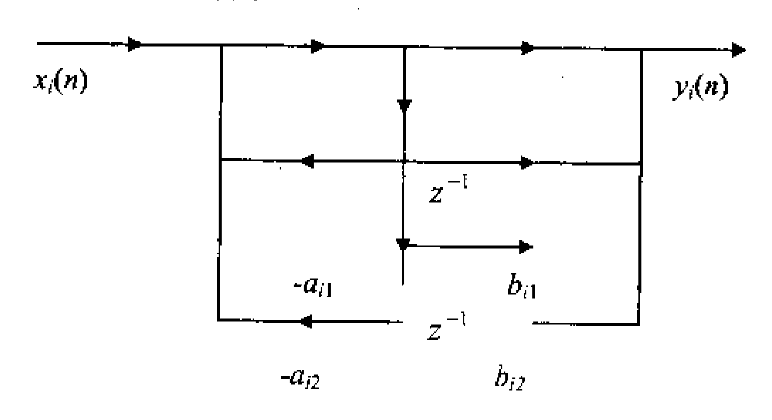


图3.3　二次分式的信号流图

总的输出：

$$y(n)=((((x(n)*h_1(n))*h_2(n))\cdots)*h_L(n)) \tag{3.2.13}$$

式中 $h_i(n)$ 是子系统 $H_i(z)$ 对应的单位抽样响应。

MATLAB 用一个 $L\times 6$ 矩阵 *sos* 来表示离散系统的二次分式，*sos* 的每一行包含一个二次分式，由 3 个分子系数和 3 个分母系数组成

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

若 N 为奇数，那么子的数目应为 $L=(N+1)/2$，其中包含一个一阶子系统。

3.2.3 离散系统的内部描述

不论是在时域还是在频域，离散系统的表示方法都是用 $h(n)$ 来表征整个系统的外部特征，而不涉及系统的内部结构。为了更好地分析与实现系统，必须了解系统的内部结构，从而引入了离散系统的内部描述，即状态空间表示方法。

一个 LSI 离散系统，可用如下的状态方程和输出方程描述：

$$\begin{aligned} w(n+1) &= Aw(n)+Bx(n) \\ y(n) &= Cw(n)+Dx(n) \end{aligned} \tag{3.2.14}$$

式中 $w(n)=[w_1(n),w_2(n),\cdots,w_N(n)]^T$ 称为系统的状态向量，$x(n)=[x_1(n),x_2(n),\cdots,x_J(n)]^T$ 是系统的输入向量，$y(n)=[y_1(n),y_2(n),\cdots,w_P(n)]^T$ 是系统的输出向量。A,B,C,D 分别是方程的系数矩阵。

若系统为单输入、单输出系统，则由状态方程可求出系统的转移函数：

$$H(z)=C[zI-A]^{-1}B+D \tag{3.2.15}$$

同样，由状态方程也可求出系统的输出及单位抽样响应 $h(n)$。

3.3 离散系统的 MATLAB 实现

如果给定一个离散系统的输入输出差分方程或系统结构，利用 MATLAB 内部函数容易求得系统的单位抽样响应 $h(n)$、频率响应及零极点增益。本节通过下面的例子来说明 MATLAB 函数的用法。

如图 3.4 所示的一个离散系统：

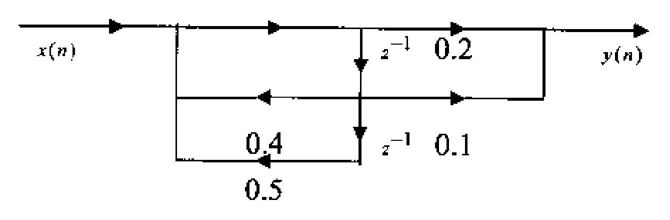


图3.4 离散系统的信号流图

该离散系统对应的输入输出差分方程为：

$$y(n)-0.4y(n-1)-0.5y(n-2)=0.2x(n)+0.1x(n-1)$$

3.3.1 单位抽样响应 $h(n)$

系统的单位抽样响应是当输入信号为单位抽样信号时系统的输出响应，有很多方法可以产生单位抽样信号，一个最直接的方法就是利用 MATLAB 中的 zeros 函数，比如产生一个 64 点的单位抽样信号：

```
pul=[1 zeros(1,63)];
```

MATLAB 中有两种函数可以计算系统的单位抽样响应：FILTER 函数和 IMPZ 函数。

1. FILTER函数

FILTER 函数是利用递归滤波器或非递归滤波器对数据进行滤波。因为一个离散系统可以看作一个滤波器，系统的输出就是输入经过滤波器滤波的结果。FILTER 函数有两种格式：

```
y=FILTER(b,a,x)
[y,zf]=FILTER(b,a,x,zi)
```

格式 y=FILTER(b,a,x)是由 b 和 a 组成的系统对输入 x 进行滤波，如果输入为单位抽样信号 $\delta(n)$，那么输出 y 就是系统的单位抽样响应 $h(n)$。

从图 3.4 中可以得到：

```
b=[0.2 0.1];
a=[1 -0.4 -0.5];
```

则系统的单位抽样响应 $h(n)$ 为：

```
h=FILTER(b,a,pul);
```

格式[y,zf]=FILTER(b,a,x,zi)是用 zi 来指定 x 的初始状态，除得到结果向量 y，还得到 x 的最终状态向量 zf。

2. IMPZ函数

IMPZ 函数只需一条语句：

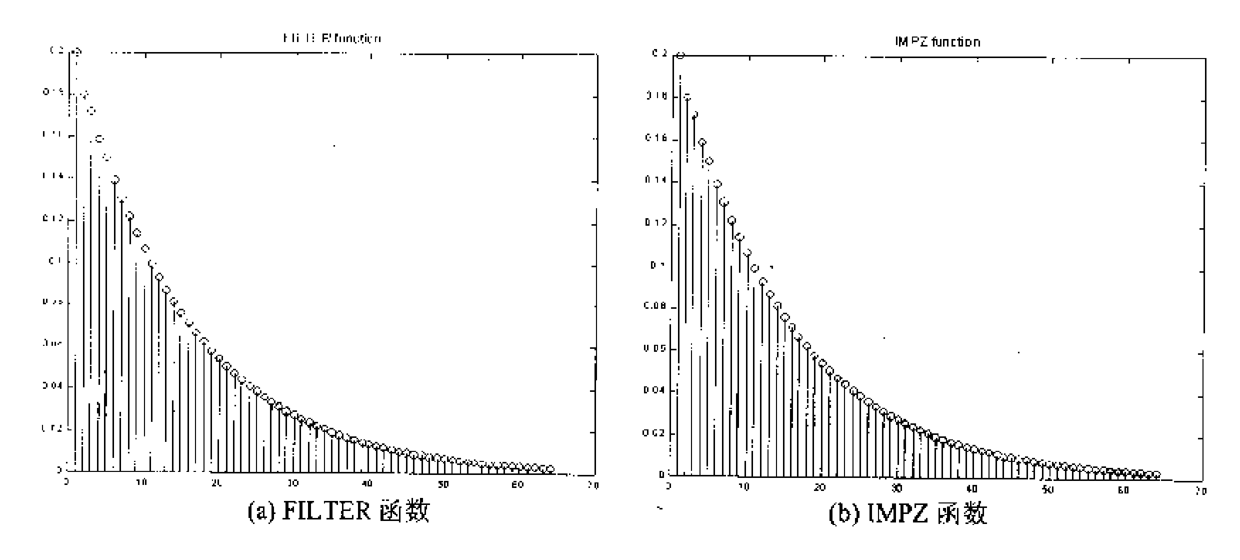
```
IMPZ(b,a);
```

便可直接给出系统的单位抽样响应，并且画出系统的单位抽样响应的图形。

下面这段程序分别利用 FILTER 函数和 IMPZ 函数得到图 3.4 所示系统的单位抽样响应曲线，如图 3.5 所示，从中可看出两函数求得的单位抽样响应相同。

```
pul=[1 zeros(1,63)];%generate unit sample sequence
b=[0.2 0.1];
a=[1 -0.4 -0.5];%b and a are system parameters
h=filter(b,a,pul);
h1=impz(b,a,64);
figure(1)
stem(h)
```

```
title('FILTER function')
figure(2)
stem(h1)
title('IMPZ function')
```



(a) FILTER 函数　　(b) IMPZ 函数

图3.5　单位抽样响应

3.3.2　频率响应 $H(e^{jw})$

MATLAB 中的 FREQZ 函数使用基于 FFT 的算法来计算由向量 a 和 b 构成的系统的频率响应。

$$H(e^{jw})=\frac{\sum_{r=0}^{M}b(r)e^{-jwr}}{1+\sum_{k=1}^{N}a(k)e^{-jwk}}$$

其一般用法为：

```
[h,f]=FREQZ(b,a,n,Fs)
```

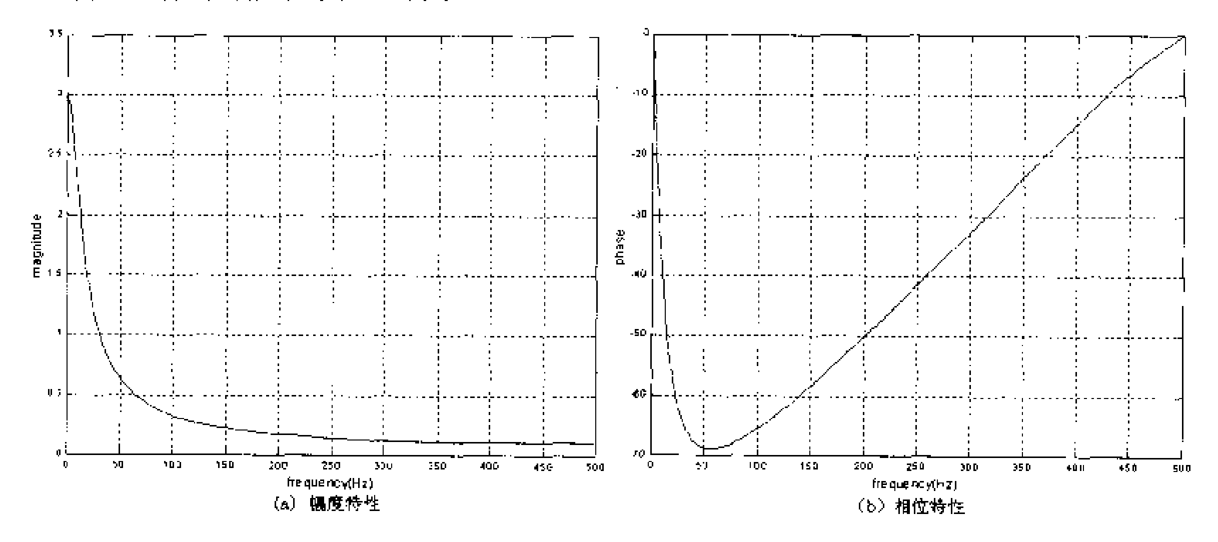
其中向量 a 和 b 由离散系统决定，允许指定采样终止频率 Fs，在[0 Fs/2]频率范围内选取 n 个频率点，并记录在 f 中。由于 FREQZ 函数是采用基 2 的 FFT 算法，n 常取 2 的幂次方，这样可以提高运算速度。

要计算系统的频率响应，可采用以下语句实现：

```
Fs=1000;%sample frequency unit in Hz
b=[0.2 0.1];
a=[1 -0.4 -0.5];
[h,f]=freqz(b,a,256,Fs);
mag=abs(h);%compute magnitude
ph=angle(h);%compute phase
ph=ph*180/pi;%unit in degrees
subplot(211),plot(f,mag);grid
xlabel('frequency(Hz)');
```

```
ylabel('magnitude');
subplot(212),plot(f,ph);grid
xlabel('frequency(Hz)');
ylabel('magnitude');
```

得到的频率响应如图 3.6 所示。



(a) 幅度特性　　(b) 相位特性

图3.6　系统的频率响应特性曲线

如果采用下述的简单语句：

```
FREQZ(b,a);
```

可以直接得到系统的幅频和相频特性曲线，如图 3.7 所示。其中幅频特性以分贝的形式给出，频率特性曲线的横轴采用的是归一化频率，即 $F_s/2=1$。

3.3.3　零极点增益

利用 MATLAB 中的 ROOTS 函数很容易求得系统的零、极点，从而得到系统的零极点增益表示。

下面的语句可以求得图 3.7 所示系统的零、极点与增益：

```
b=[0.2 0.1];
a=[1 -0.4 -0.5];
zr=ROOTS(b)
pk=ROOTS(a)
g=b(1)/a(1)
```

所得结果为：

```
zr =
    -0.5000
pk =
    0.9348
    -0.5348
g =
```

```
    0.2000
```

利用 MATLAB 中的 ZPLANE 函数可使上面的过程简化：

```
b=[0.2 0.1];
a=[1 -0.4 -0.5];
g=b(1)/a(1);
ZPLANE(b,a);
```

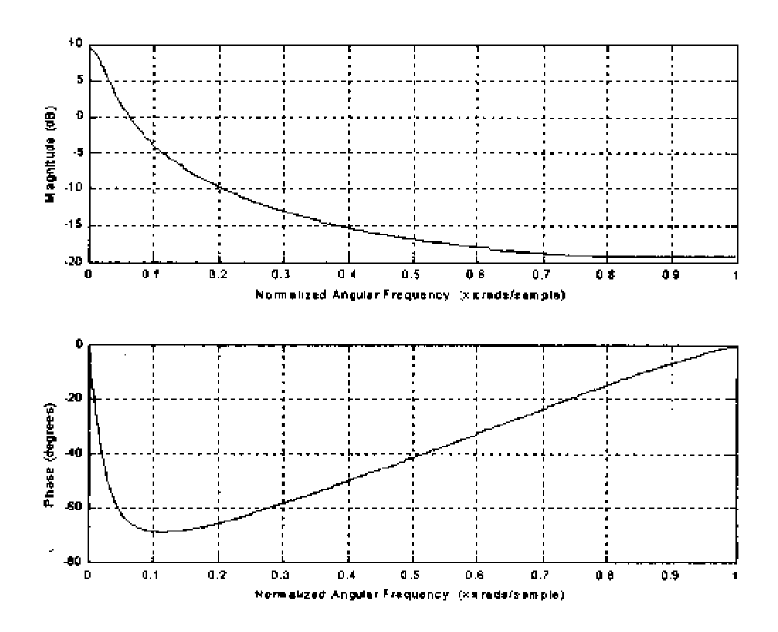


图3.7　系统的频率响应特性曲线

在这种情况下，ZPLANE 用 ROOTS 函数求出 b,a 系统的根，并画出如图 3.8 所示的零极点图。

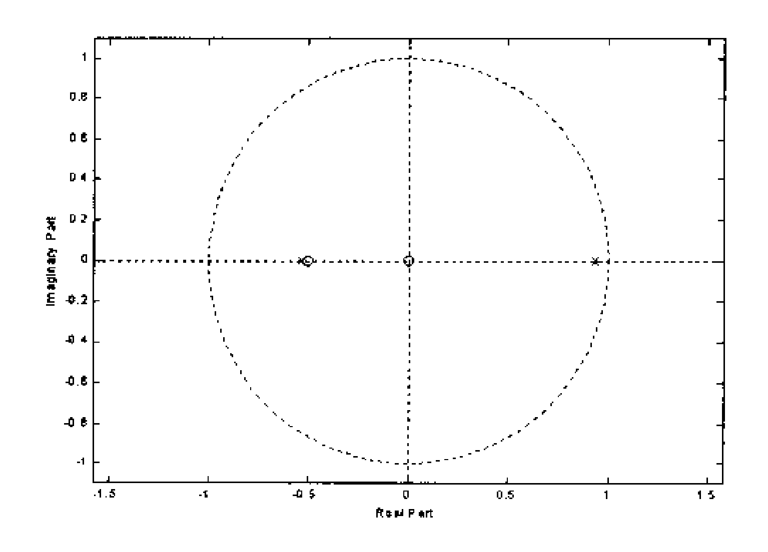


图3.8　系统的零极点

3.4　离散系统变换

同一个离散系统，可以用转移函数、零极点增益、二次分式及状态空间多种形式表示，MATLAB 工具箱中提供了很多函数来实现不同表示方法之间的转换，如表 3.1 所示。

本部分结合具体的例子说明各函数的用法。

表 3.1　离散系统变换函数

原　型	转 换 型	函　数
转移函数	零极点增益	tf2zp
转移函数	状态空间	tf2ss
零极点增益	转移函数	zp2tf
零极点增益	二次分式	zp2sos
零极点增益	状态空间	zp2ss
二次分式	转移函数	sos2tf
二次分式	零极点增益	sos2zp
二次分式	状态空间	sos2ss
状态空间	转移函数	ss2tf
状态空间	零极点增益	ss2zp
状态空间	二次分式	ss2sos

1. tf2zp函数

tf2zp 函数变系统转移函数形式：

$$H(z)=\frac{\sum_{r=0}^{M}b(r)z^{-r}}{1+\sum_{k=1}^{N}a(k)z^{-k}}$$

为零极点增益形式：

$$H(z)=g\frac{\prod_{r=1}^{M}(z-z_r)}{\prod_{k=1}^{N}(z-p_k)}$$

tf2zp 函数的格式为：

```
[z,p,g]=tf2zp(b,a)
```

例如，要找出系统

$$H(z)=\frac{0.5z^{-1}+0.3z^{-2}+0.2z^{-3}}{1+0.2z^{-1}+0.4z^{-2}-0.8z^{-3}}$$

的零、极点和增益，并画出零极图，则可以通过下面的语句实现：

```
b=[0 0.5 0.3 0.2];
a=[1 0.2 0.4 0.8];
     [z,p,g]=tf2zp(b,a);
     zplane(z,p)
```

输出结果如下：

```
z =
   -0.3000 + 0.5568i
   -0.3000 - 0.5568i
p =
    0.3228 + 0.9175i
    0.3228 - 0.9175i
   -0.8457
g =
5000
```

得到如图 3.9 所示的零极图。

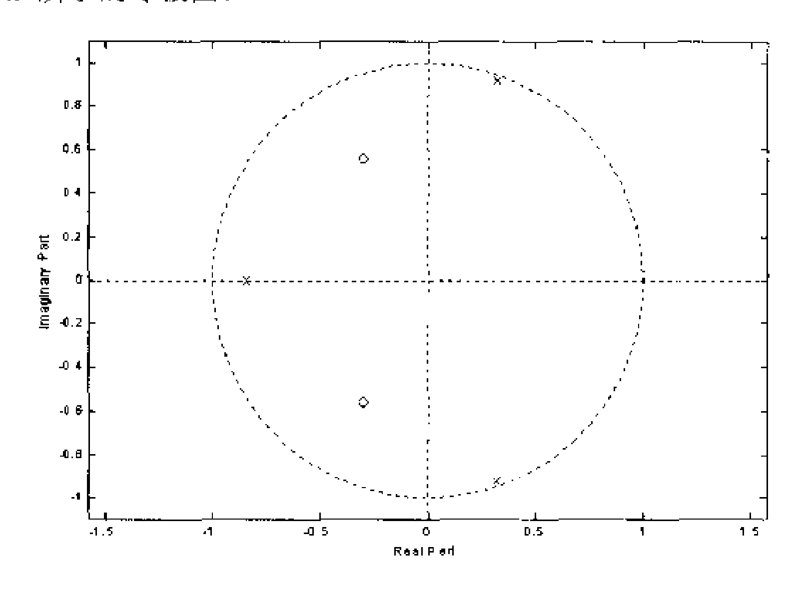


图3.9　系统的零极图

2.　tf2ss函数

tf2ss 函数变系统转移函数形式：

$$H(z)=\frac{\sum_{r=0}^{M} b(r)z^{-r}}{1+\sum_{k=1}^{N} a(k)z^{-k}}$$

为状态空间形式：

$$w(n+1)=Aw(n)+Bx(n)$$
$$y(n)=Cw(n)+Dx(n)$$

tf2ss 函数的格式为：

```
[A,B,C,D]=tf2ss(b,a)
```

例如，将系统：

$$H(z)=\frac{1+4z^{-1}+4z^{-2}}{1+0.6z^{-1}+z^{-2}}$$

变换成状态空间表示：

```
b=[1 4 4];
a=[1 0.6 1];
[A,B,C,D]=tf2ss(b,a)
```

输出结果如下：

```
A =
   -0.6000   -1.0000
    1.0000         0
B =
    1
    0
C =
    3.4000    3.0000
D =
    1
```

3. zp2tf函数

zp2tf 函数是 tf2zp 函数的反变换形式，将系统零极点增益形式变换为转移函数形式。其格式为：

```
[b,a]=zp2tf(z,p,g)
```

这里以 tf2zp 函数所得结果为例，输入语句：

```
z =[-0.3000 + 0.5568i ;  -0.3000 - 0.5568i];
p =[0.3228 + 0.9175i  ;  C.3228 - 0.9175i  ;  -0.8457];
g =0.5000;
[b,a]=zp2tf(z,p,g)
```

输出结果如下：

```
b =
    0         0.5000    0.3000    0.2000
a =
   1.0000    0.2001    0.4000    0.8000
```

4. zp2sos函数

zp2sos 函数变系统零极点增益形式：

$$H(z)=g\frac{\prod_{r=1}^{M}(z-z_r)}{\prod_{k=1}^{N}(z-p_k)}$$

为二次分式形式：

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

zp2sos 函数的格式为：

```
sos=zp2sos(z,p,k,'order')
```

order 用于指定 sos 中的行顺序，order 为 down 时，表示 sos 中第 1 行所包含的极点离单位圆最近；反之，order 为 up 时，表示 sos 中第 1 行所包含的极点离单位圆最远。

例如，下面语句实现系统零极点增益形式变二次分式形式。

```
z=[0.56;1.42+0.25i; 1.42-0.25i];
p=[0.5;0.8;-0.2;-0.4];
g=0.5;
sos=zp2sos(z,p,g)
```

输出结果如下：

```
sos =
        0      0.5000   -0.2800    1.0000    0.6000    0.0800
   1.0000   -2.8400    2.0789    1.0000   -1.3000    0.4000
```

5. zp2ss函数

zp2ss 函数变系统零极点增益形式：

$$H(z)=g\frac{\prod_{r=1}^{M}(z-z_r)}{\prod_{k=1}^{N}(z-p_k)}$$

为状态空间形式：

$$w(n+1)=Aw(n)+Bx(n)$$
$$y(n)=Cw(n)+Dx(n)$$

zp2ss 函数的格式：

```
[A,B,C,D]=zp2ss(z,p,g)
```

例如，输入下面语句：

```
z=[0.56;1.42+0.25i; 1.42-0.25i];
p=[0.5;0.8;-0.2;-0.4];
k=0.5;
[A,B,C,D]=zp2ss(z,p,g)
```

输出结果如下：

```
A =
    1.3000   -0.6325        0         0
    0.6325        0         0         0
    1.0000   -0.8854   -0.6000   -0.2828
         0        0     0.2828        0
B =
    1
    0
    0
    0
C =
    0.5000   -0.4427   -1.7200    3.5336
D =
   0
```

6.　sos2tf函数

sos2tf 函数变系统二次分式形式：

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

为转移函数形式：

$$H(z)=\frac{\sum_{r=0}^{M} b(r)z^{-r}}{1+\sum_{k=1}^{N} a(k)z^{-k}}$$

sos2tf 函数的格式为：

```
[b,a]=sos2tf(sos)
```

例如，已知一系统的二次分式表示形式为：

$$sos=\begin{bmatrix} 1 & 3 & 2 & -2 & 0 & 1 \\ -2 & 1 & 1 & 4 & 6 & 3 \\ -3 & -2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & -2 & -1 & -3 \\ 1 & -1 & 1 & -1 & -1 & 2 \end{bmatrix}$$

要将该系统变为转移函数表示形式。

实现程序：

```
sos=[1  3   2   -2  0   1
    -2  1   1   4   6   3
    -3  -2  1   0   5   4
```

```
    4   1   2   -2  -1  -3
    1   -1  1   -1  -1  2];
[b,a]=sos2tf(sos)
```

输出结果如下：

```
b =
    24   58   5   -15   29   -18   -39   -25   -23   0   4
a =
    0   -80   -304   -392   -170   292   635   406   -93   -222   -72
```

7. sos2zp函数

sos2zp 函数变系统二次分式形式：

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

为零极点增益形式：

$$H(z)=g\frac{\prod_{r=1}^{M}(z-z_r)}{\prod_{k=1}^{N}(z-p_k)}$$

sos2zp 函数的格式为：

```
[z,p,g]=sos2zp(sos)
```

例如，已知一系统的二次分式表示形式为：

$$sos=\begin{bmatrix} 0 & 0.5 & -0.3 & 1 & 0.6 & 0.1 \\ 1 & -2.8 & 2 & 1 & -1.3 & 0.4 \\ 2.5 & 1 & -0.5 & 0.4 & 1.2 & 1.6 \\ -1.4 & 0.3 & 0.6 & -0.7 & 0 & 0.9 \end{bmatrix}$$

要将该系统用零极点增益形式表示。

实现上述功能，输入：

```
sos=[0      0.5    -0.3    1      0.6    0.1
     1      -2.8   2       1      -1.3   0.4
     1.5    1      -0.5    0.4    1.2    1.6
    -1.4    0.3    0.6    -0.7    0      0.9];
[z,p,g]=sos2zp(sos)
```

输出结果如下：

```
z =
    0.6000
    1.4000 + 0.2000i
    1.4000 - 0.2000i
   -0.6899
```

```
     0.2899
     0.7705
    -0.5562
p =
   -0.3000 + 0.1000i
   -0.3000 - 0.1000i
   0.8000
   0.5000
   -1.5000 + 1.3229i
   -1.5000 - 1.3229i
    1.1339
   -1.1339
g =
   6.2500
```

8. sos2ss函数

sos2ss 函数变系统二次分式形式：

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

为状态空间形式：

$$w(n+1)=Aw(n)+Bx(n)$$
$$y(n)=Cw(n)+Dx(n)$$

sos2ss 函数的格式为：

```
[A,B,C,D]=sos2ss(sos)
```

例如，用状态空间形式表示系统：

$$sos=\begin{bmatrix} 1 & 3 & 2 & -2 & 1 & 1 \\ -2 & 1 & 1 & 4 & 6 & 3 \\ -3 & -2 & 1 & 1 & 5 & 4 \\ 4 & 1 & 2 & -2 & -1 & -3 \\ 1 & -1 & 1 & -1 & -1 & 2 \end{bmatrix}$$

输入程序：

```
sos=[1   3  2   -2  1   1
    -2   1  1    4  6   3
    -3  -2  1    1  5   4
     4   1  2   -2  -1  -3
     1  -1  1   -1  -1   2];
[A,B,C,D]=sos2ss(sos)
```

输出结果如下：

```
A =
  Columns 1 through 7
```

```
   -7.5000   -17.5000   -13.6250    7.4375   32.1250  36.3750  7.3750
    1.0000         0          0          0         0        0       0
         0    1.0000          0          0         0        0       0
         0         0     1.0000          0         0        0       0
         0         0          0     1.0000         0        0       0
         0         0          0          0    1.0000        0       0
         0         0          0          0         0   1.0000       0
         0         0          0          0         0        0  1.0000
         0         0          0          0         0        0       0
         0         0          0          0         0        0       0
  Columns 8 through 10
  -20.8125  -18.3750   -4.5000
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    1.0000         0         0
         0    1.0000         0
B =
     1
     0
     0
     0
     0
     0
     0
     0
     0
     0
C =
  Columns 1 through 7
   7.6250   25.9375   21.3750  -12.9688  -47.0625  -52.1250   -9.5000
   Columns 8 through 10
   32.6563   27.5625    6.5000
D =
   -1.5000
```

9. ss2tf函数

ss2tf 函数是 tf2ss 函数的逆变换，变系统状态空间形式为转移函数形式。

ss2tf 函数的格式为：

```
[b,a]=ss2tf(A,B,C,D,iu)
```

其中，iu 用于指定所使用的输入量。

这里以 tf2ss 函数所得结果为例，输入语句：

```
A =[-0.6  -1;1  0];
```

```
B =[1;0];
C =[3.4 3];
D =1;
[b,a]=ss2tf(A,B,C,D,1)
```

输出结果如下：

```
b =
     1     4     4
a =
   1.0000    0.6000    1.0000
```

10. ss2zp函数

ss2zp 函数变系统状态空间形式为零极点增益形式，格式为：

```
[z,p,g]=ss2zp(A,B,C,D,iu)
```

它是 zp2ss 函数的逆变换，iu 的说明可以参照 ss2tf 函数。

例如，输入语句：

```
A=[1.3000   -0.6325     0          0
   0.6325       0       0          0
   1.0000   -0.8854  -0.6000   -0.2828
     0          0     0.2828       0];
B =[1;0;0;0];
C =[0.5000   -0.4427   -1.7200    3.5336];
D =0;
[z,p,g]=ss2zp(A,B,C,D,1)
```

输出结果如下：

```
z =
    1.4200 + 0.2494i
    1.4200 - 0.2494i
    0.5600
p =
   0.7998
   0.5002
   -0.4001
   -0.1999
g =
   5000
```

11. ss2sos函数

ss2sos 函数变系统状态空间形式：

$$w(n+1) = Aw(n) + Bx(n)$$
$$y(n) = Cw(n) + Dx(n)$$

为二次分式形式：

$$sos=\begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

ss2sos 函数的格式为

```
sos=ss2sos(A,B,C,D,iu,'order')
```

对于多输入系统，iu 用于指定变换中所使用的输入量。order 用于指定 sos 中的行顺序，order 为 down 时，表示 sos 中第 1 行所包含的极点离单位圆最近，反之，order 为 up 时，表示 sos 中第 1 行所包含的极点离单位圆最远。

在举例说明以前，先介绍一个可以产生状态方程的函数：besself 函数。其格式为：

```
[A,B,C,D]=besself(n,Wn)
```

besself 函数用于设计模拟滤波器，n 为滤波器的阶数，Wn 为截止频率。

例如，利用 besself 函数产生系统的状态方程，然后由 ss2sos 函数求出二次分式形式。

输入程序：

```
[A,B,C,D]=besself(4,0.6)
sos=ss2sos(A,B,C,D)
```

输出结果如下：

```
A =
    -1.0857     -0.5667      0            0
     0.5667        0         0            0
        0        0.6353   -0.7887    -0.6353
        0          0       0.6353       0
B =
   0.6000
   0
   0
   0
C =
   0         0         0    0.9444
D =
   0
sos =
     0         0    0.1296    1.0000    1.0857    0.3211
     0         0    1.0000    1.0000    0.7887    0.4036
```

第4章　信号变换及其MATLAB实现

MATLAB中有如下几种变换函数：

- 离散傅立叶变换DFT。即单位圆上的z变换，此变换使离散序列的描述和处理变得容易。
- Chirp z 变换。此变换在沿轮廓线而非单位圆的 z 变换中非常有用，特别是计算prime-length变换时，Chirp z变换比DFT算法效率更高。
- 离散余弦变换 DCT。此变换与 DFT 的关系很接近，其能量压缩特性在信号代码应用方面非常有用。
- Hilbert变换。能简化解析信号的格式，在通讯领域的带通信号处理方面有着广泛的作用。

另外，Z变换是离散系统与离散信号分析与综合的重要工具，且通过MATLAB很容易实现序列的Z变换及其特性。

4.1　离散傅立叶变换

有限长序列作为离散信号的一种，在数字信号处理中占有很重要的作用。对于有限长序列，傅立叶变换不仅在理论上有重要的意义，而且在各种数字信号处理的运算方法中，越来越起到核心的作用。

为了便于更好的理解DFT的概念，我们先简要介绍一下周期序列与离散傅立叶级数。

4.1.1　周期序列与傅立叶级数

我们用$\tilde{x}(n)$表示周期为N的周期序列，即：

$$\tilde{x}(n)=\tilde{x}(n+kN) \qquad k\text{为任意整数}$$

周期序列不能进行z变换，因为在z平面上没有任何收敛区域。但是，周期序列可以用离散的傅立叶级数来表达，也就是用周期为N的正弦序列来表示。

周期为N的正弦序列其基频成分为：

$$e_1(n)=e^{j(2\pi/N)n}$$

其k次谐波序列为：

$$e_k(n)=e^{j(2\pi/N)kn}$$

即

$$e_{k+N}(n)=e_k(n)$$

也就是说，离散级数所有谐波成分中只有 N 个是独立的，因而在展成离散傅立叶级数时，只能取 k=0 到 $N-1$ 为止的 N 个独立的谐波分量，否则将出现二义性。由此得到如下的离散傅立叶级数公式：

$$\widetilde{x}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\widetilde{X}(k)e^{j(2\pi/N)kn} \tag{4.1.1}$$

式中求和号前所乘的系数 $\frac{1}{N}$ 是习惯上已经采用的常数，$\widetilde{X}(k)$ 是 k 次谐波的系数：

$$\widetilde{X}(k)=\sum_{n=0}^{N-1}\widetilde{x}(n)e^{-j(2\pi/N)kn} \tag{4.1.2}$$

式(4.1.1)和(4.1.2)称为周期序列的离散傅立叶级数表示。

习惯上也常采用符号 W_N：

$$W_N=e^{-j(2\pi/N)}$$

这样，离散傅立叶级数对可表示为

$$\begin{cases}\widetilde{X}(k)=\sum\limits_{n=0}^{N-1}\widetilde{x}(n)W_N^{kn}\\ \widetilde{x}(n)=\frac{1}{N}\sum\limits_{k=0}^{N-1}\widetilde{X}(k)W_N^{-kn}\end{cases} \tag{4.1.3}$$

4.1.2 离散傅立叶变换 DFT

周期序列实际上只有有限个序列值有意义，因此它的许多特性可以衍用到有限长序列上。对于一个长度为 N 的有限长序列 $x(n)$，也即 $x(n)$ 只在 $n=0$ 到 $(N-1)$ 个点上有非零值，其余皆为零，即

$$x(n)=\begin{cases}x(n) & 0\leqslant n\leqslant N-1\\ 0 & 其余\ n\end{cases} \tag{4.1.4}$$

以序列 $x(n)$ 为主值序列并以 N 为周期延拓得到周期序列 $\widetilde{x}(n)$，因此它们的关系为：

$$\widetilde{x}(n)=\sum_{r=-\infty}^{\infty}x(n+rN) \tag{4.1.5}$$

$$x(n)=\begin{cases}\widetilde{x}(n) & 0\leqslant n\leqslant N-1\\ 0 & 其余\ n\end{cases} \tag{4.1.6}$$

由离散傅立叶级数对公式(4.1.3)，我们很容易得到有限长序列 $x(n)$ 的离散傅立叶变换公式：

$$\begin{cases} X(k)=\sum_{n=0}^{N-1} x(n)W_N^{kn} & 0 \leqslant k \leqslant N-1 \\ x(n)=\dfrac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn} & 0 \leqslant n \leqslant N-1 \end{cases} \tag{4.1.7}$$

例，若 $x(n)=\cos(\frac{n\pi}{6})$ 是一个 $N=12$ 的有限序列，利用 MATLAB 计算它的 DFT 并画出图形。

程序清单：

```
N=12;%length of sequence
n=0:N-1;%time sample
xn=cos(pi*n/6);%generate sequence
k=0:N-1;%frequency sample
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=xn*WNnk %compute DFT
%plot sequence and its DFT
figure(1)
stem(n,xn)
figure(2)
stem(k,abs(Xk))
```

有限序列的 DFT 为：

```
Xk =
   Columns 1 through 4
   -0.0000  6.0000 + 0.0000i  -0.0000 - 0.0000i  -0.0000 - 0.0000i
   Columns 5 through 8
   -0.0000-0.0000i  -0.0000-0.0000i  -0.0000-0.0000i   0.0000-0.0000i
   Columns 9 through 12
   0.0000-0.0000i   0.0000-0.0000i   0.0000-0.0000i   6.0000 + 0.0000i
```

有限序列及其 DFT 的图形如图 4.1 和 4.2 所示。

同样，由离散傅立叶反变换公式，利用 MATLAB 容易编出计算程序，这里提供给读者。

```
function xn=IDFT(Xk,N)
%this program computes Inverse Discrete Fourier Transform
%its format is
%xn= IDFT(Xk,N)
%N is length of IDFT
%Xk is DFT coeff. Array over 0<=k<=N-1
%xn is N-point sequence over 0<=k<=N-1
n=0:N-1;
k=0:N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^(-nk);
```

```
xn=(Xk*WNnk)/N; %compute IDFT
%plot input Xk and its IDFT
subplot(211),stem(k,abs(Xk));
xlabel('k');
ylabel('abs(Xk)');
subplot(212),stem(n,real(xn))
xlabel('n');
ylabel('real(xn)');
```

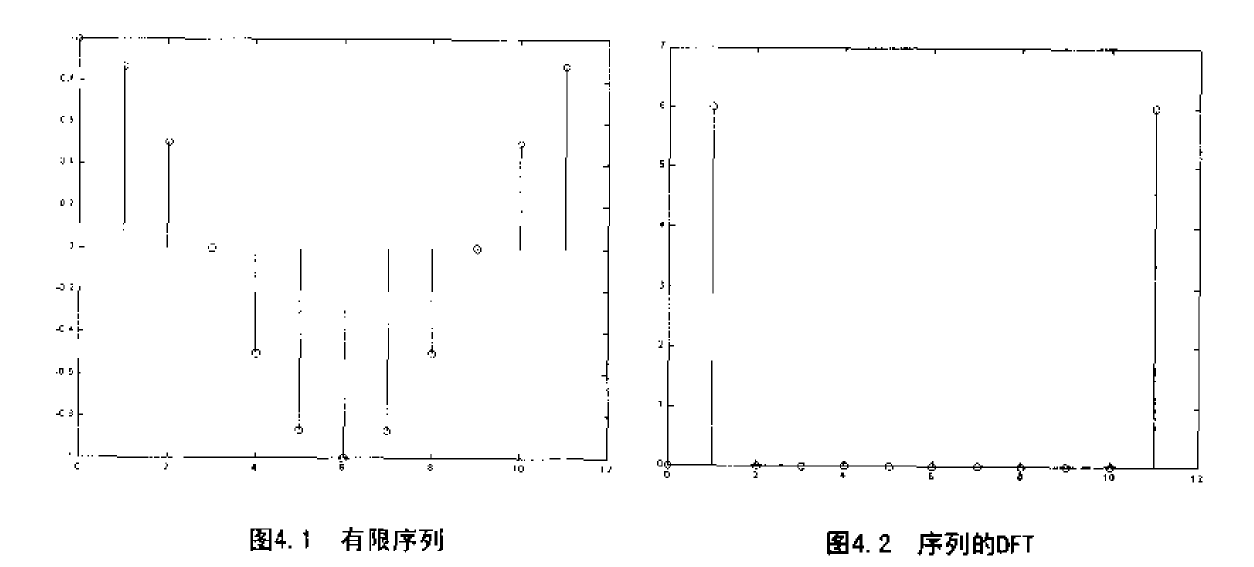


图4.1　有限序列　　　　图4.2　序列的DFT

4.1.3　DFT 的性质

1.　线性

若 $x_1(n)$，$x_2(n)$ 都是 N 点序列，其 DFT 分别是 $X_1(k)$，$X_2(k)$，则：

$$\mathrm{DFT}[ax_1(n)+bx_2(n)]=aX_1(k)+bX_2(k) \tag{4.1.8}$$

2.　正交性

令矩阵

$$W_N=[W^{nk}]=\begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)(N-1)} \end{bmatrix} \tag{4.1.9}$$

$$X_N=[X(0),X(1),\cdots,X(N-1)]^{\mathrm{T}}$$

$$x_N=[x(0),x(1),\cdots,x(N-1)]^{\mathrm{T}}$$

则 DFT 的正变换可写成矩阵形式：

$$X_N=W_N x_N \tag{4.1.10}$$

由于

$$W_N^* W_N = \sum_{k=0}^{N-1} W^{mk} W^{-nk} = \begin{cases} N & m=n \\ 0 & m \neq n \end{cases} \tag{4.1.11}$$

所以 W_N^* 和 W_N 是正交的，也即 W_N 是正交矩阵，DFT 是正交变换。进一步，有

$$W_N^* W_N = NI \tag{4.1.12}$$

DFT 的逆变换可表示为：

$$x_N = W_N^{-1} X_N = \frac{1}{N} W_N^* X_N \tag{4.1.13}$$

由于 MATLAB 在矩阵操作方面具有独到的优越性，所以利用 DFT 的正交性，可以通过下面简单的语句实现 DFT 与 IDFT：

```
n=0:N-1;
k=0:N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
IWNnk=WN.^(-nk);
Xk=xn*WNnk;
xn=(Xk*IWNnk)/N;
```

3. 圆周移位

一个有限长序列 $x(n)$ 的圆周移位定义为：

$$f(n) = x((n-m))_N R_N(n) \tag{4.1.13}$$
$$x((n-m))_N = \tilde{x}(n-m)$$
$$R_N(n) = \begin{cases} 1 & 0 \leqslant n \leqslant N-1 \\ 0 & 其余\ n \end{cases}$$

其中 $x((n-m))_N$ 称为序列 $x(n)$ 的周期延拓。

序列 $x(n)$ 圆周移位后的 DFT 为：

$$\mathrm{DFT}[f(n)] = W_N^{mk} X(k) \tag{4.1.14}$$

实现序列的圆周移位，在 MATLAB 中可以通过 mod 函数实现。下面通过具体的例子来说明。

例如，求有限长序列 $x(n) = 5(0.6)^n \quad 0 \leqslant n < 20$

的圆周移位 $f(n) = x((n-10))_{20} R_{20}(n)$。

输入下列程序：

```
%example figure 4.3
function ex43
N=20;%length of sequence
m=10;%shift length
n=0:N-1;%sample point
x=20*(0.6).^n%generate original sequence
%compute circular shift sequence
%Method:f(n)=(x(n-m)mod N)
%f is sequence after circular shift
n1=mod((n-m),N);
```

```
f=x(n1+1)
%plot the original and circular shift sequences
subplot(211),stem(n,x);
title('original sequence');
xlabel('n');
ylabel('x(n)');
subplot(212),stem(n,y);
title('circular shift sequence');
xlabel('n');
ylabel('x((n-10)mod 20)');
```

输出结果：

```
x =
    20.0000   12.0000    7.2000    4.3200    2.5920    1.5552    0.9331
     0.5599    0.3359    0.2016    0.1209    0.0726    0.0435    0.0261
     0.0157    0.0094    0.0056    0.0034    0.0020    0.0012
f =
    0.1209    0.0726    0.0435    0.0261    0.0157    0.0094    0.0056
    0.0034    0.0020    0.0012   20.0000   12.0000    7.2000    4.3200
    2.5920    1.5552    0.9331    0.5599    0.3359    0.2016
```

得到如图 4.3 所示的曲线。

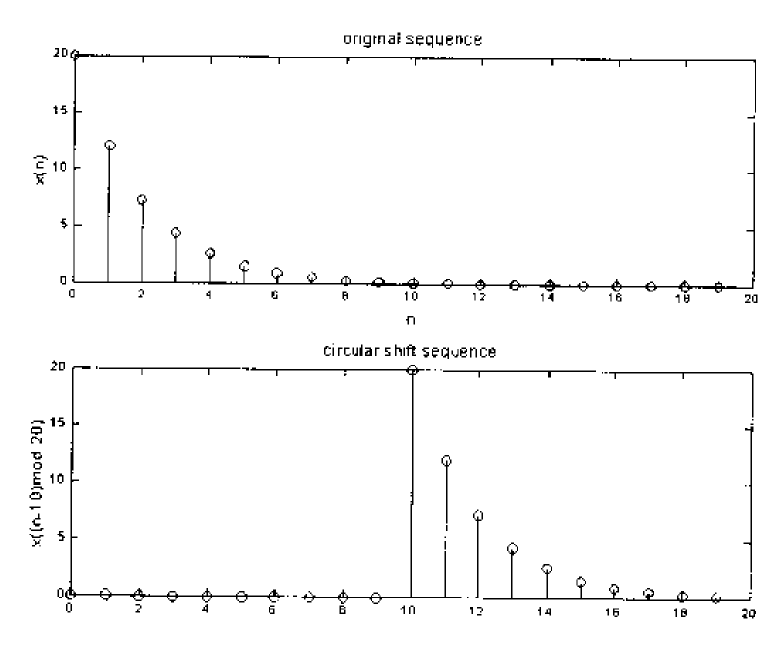


图4.3　序列的圆周移位

4.　圆周卷积

若 $F(k)=X(k)Y(k)$，

则
$$f(n)=\mathrm{IDFT}[F(k)]=\sum_{m=0}^{N-1}x(m)y((n-m))_N R_N(n)$$

或
$$f(n)=\mathrm{IDFT}[F(k)]=\sum_{m=0}^{N-1}y(m)x((n-m))_N R_N(n) \tag{4.1.15}$$

基于 DFT 的圆周移位特性，在 MATLAB 中可构造计算圆周卷积的函数，程序如下：

```
function [y,n]=circonv(x1,x2,N)
% implement circle convolution of x1 and x2
% x1 and x2 are input sequences of n1,n2<=N
% y is output sequence
% N is size of circle buffer
% Method:y(n)=sum(x1(m)*x2((n-m) mod N))
% check for length of x1
if(length(x1)>N|length(x2)>N)
 error('N must be >= length(x1)')
end
x1=[x1 zeros(1,N-length(x1))];
x2=[x2 zeros(1,N-length(x2))];
for n=1:N
 for m=1:N
   p=n-m;
   y(n,m)=x1(m)*x2(mod(p,N)+1);
 end
end
y=sum(y');
```

下面通过例子说明。

例，计算序列 $x_1(n)=(0.6)^n$ $(0\leqslant n<10)$与 $x_2(n)=(0.4)^n(0\leqslant n<15)$的圆周卷积$(N=20)$。

实现程序：

```
%example figure 4.4
function ex44
N=20;%length of circular convolution
N1=10;%length of sequence x1(n)
N2=15;%length of sequence x2(n)
n1=0:N1-1;
n2=0:N2-1;
x1=(0.6).^n1
x2=(0.4).^n2
y=circonv(x1,x2,N)
subplot(311),stem(n1,x1)
xlabel('n1');ylabel('x1(n1)');
subplot(312),stem(n2,x2)
xlabel('n2');ylabel('x2(n2)');
n=0:N-1;
subplot(313),stem(n,y)
xlabel('n');ylabel('y(n)');
```

输出结果：

```
x1 =
    1.0000  0.6000  0.3600  0.2160  0.1296  0.0778  0.0467
    0.0280  0.0168  0.0101
x2 =
    1.0000  0.4000  0.1600  0.0640  0.0256  0.0102  0.0041  0.0016
    0.0007  0.0003  0.0001  0.0000  0.0000  0.0000  0.0000
```

```
y =
    1.0000  1.0000  0.7600  0.5200  0.3376  0.2128  0.1318  0.0807
    0.0491  0.0297  0.0119  0.0048  0.0019  0.0008  0.0003  0.0001
    0.0000  0.0000  0.0000  0.0000
```

得到如图 4.4 所示的曲线。

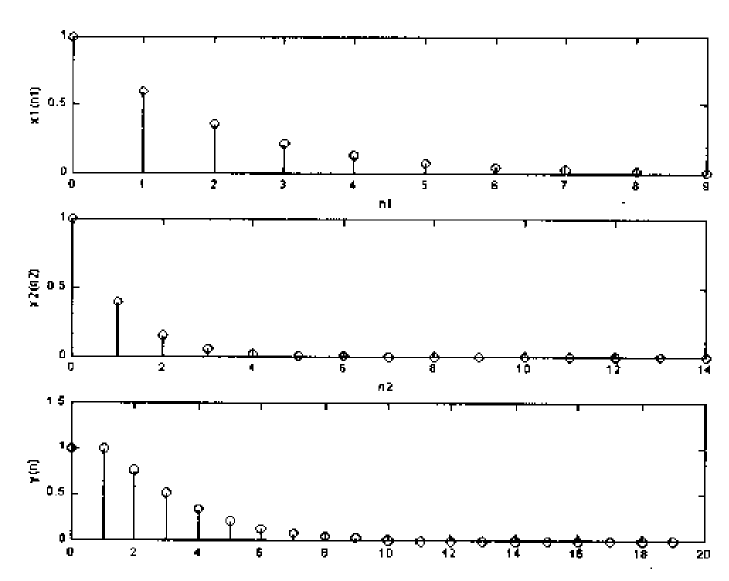


图4.4 序列的圆周卷积

5. **共轭对称性**

设 $x^*(n)$ 为 $x(n)$ 的共轭复数序列，则

$$\mathrm{DFT}[x^*(n)] = X^*((N-k))_N \tag{4.1.16}$$

以 $x_r(n)$ 及 $x_i(n)$ 分别表示序列 $x(n)$ 的实部与虚部，即

$$x(n) = x_r(n) + jx_i(n)$$

$$\begin{cases} x_r(n) = \dfrac{1}{2}[x(n) + x^*(n)] \\ jx_i(n) = \dfrac{1}{2}[x(n) - x^*(n)] \end{cases} \tag{4.1.17}$$

以 $X_e(k)$ 及 $X_o(k)$ 分别表示实部及虚部序列的 DFT，即

$$\begin{aligned} X_e(k) &= \mathrm{DFT}[\,x_r(n)\,] \\ &= \frac{1}{2}[X(k) + X^*((N-k))_N] \end{aligned} \tag{4.1.18}$$

$$\begin{aligned} X_o(k) &= \mathrm{DFT}[\,x_i(n)\,] \\ &= \frac{1}{2}[X(k) - X^*((N-k))_N] \end{aligned} \tag{4.1.19}$$

并且可以证明

$$X_e(k) = X_e^*((N-k))_N \tag{4.1.20}$$

$$X_o(k) = -X_o^*((N-k))_N \tag{4.1.21}$$

通常称 $X_e(k)$ 为 $X(k)$ 的共轭偶部，$X_o(k)$ 为 $X(k)$ 的共轭奇部。所以说，对于时域、频

域的 DFT 对应关系来说，序列 $x(n)$ 的实部对应于 $X(k)$ 的共轭偶部，序列 $x(n)$ 的虚部对应于 $X(k)$ 的共轭奇部。

对于实序列来说，由上面的理论可知存在下述关系：

$$X(k)=X_e(k)=X^*((N-k))_N \tag{4.1.22}$$

另外，实序列可分解为奇偶分量：

$$\begin{cases} x_e(n)=\dfrac{1}{2}[x(n)+x(N-n)] \\ x_o(n)=\dfrac{1}{2}[x(n)-x(N-n)] \end{cases} \tag{4.1.23}$$

不难证明，其相应的 DFT 为：

$$\begin{aligned} &\mathrm{DFT}[x_e(n)]=\mathrm{Re}[X(k)] \\ &\mathrm{DFT}[x_o(n)]=\mathrm{jIm}[X(k)] \end{aligned} \tag{4.1.24}$$

在 MATLAB 中，可利用 mod 函数实现实序列分解成奇偶分量。

例，对于实序列 $x(n)=(0.9)^n \quad 0\leqslant n<20$，

(1) 分解成偶部 $x_e(n)$ 和奇部 $x_o(n)$；

(2) 验证实序列的性质：

$$\mathrm{DFT}[x_e(n)]=\mathrm{Re}[X(k)]$$
$$\mathrm{DFT}[x_o(n)]=\mathrm{jIm}[X(k)]$$

MATLAB 程序如下：

```
%example figure 4.5 and 4.6
function ex45
N=20;
n=0:N-1;
x=(0.9).^n;
%part1 decomposite x(n) into xe(n) and xo(n)
n1=mod(N-n,N);
xe=(x+x(n1+1))/2;
xo=(x-x(n1+1))/2;
figure(1)
subplot(311),stem(n,x);
xlabel('n');ylabel('x(n)');
subplot(312),stem(n,xe);
xlabel('n');ylabel('xe(n)');
subplot(313),stem(n,xo);
xlabel('n');ylabel('xo(n)');
%part2 verify property
%First compute X(k)
k=0:N-1;
nk=n'*k;
WN=exp(-j*2*pi/N);
WNnk=WN.^nk;
Xk=x*WNnk;
rXk=real(Xk)%real part of Xk
iXk=imag(Xk)%imaginary part of Xk
%Second compute Xe(k) and Xo(k)
```

```
Xek=xe*WNnk
Xok=xo*WNnk
%Third plot Re[X(k)]|Im[X(k)]|Re[Xe(k)]|Im[Xo(k)]
figure(2)
subplot(221),stem(k,rXk);
xlabel('k'); title('Re[X(k)]');
subplot(222),stem(k,iXk);
xlabel('k'); title('Im[X(k)]');
subplot(223),stem(k,real(Xek));
xlabel('k'); title('Re[Xe(k)]');
subplot(224),stem(k,imag(Xok));
xlabel('k');
title('Im[X(k)]');
```

执行后得到如图 4.5 和如图 4.6 所示的曲线。

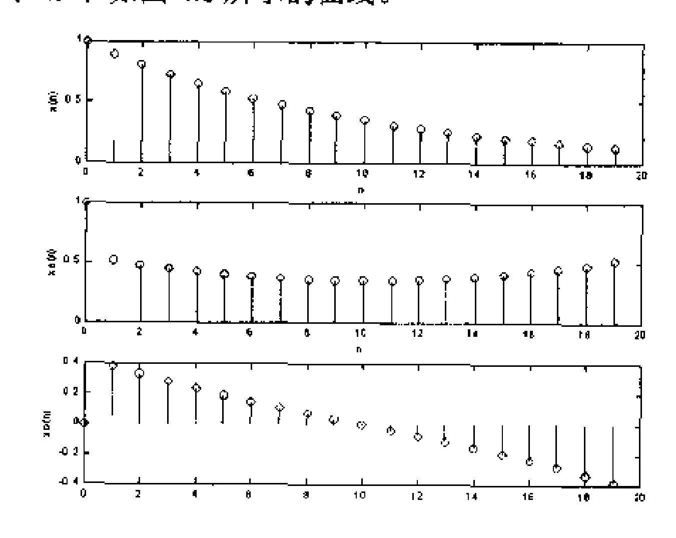


图4.5 序列的奇偶分解

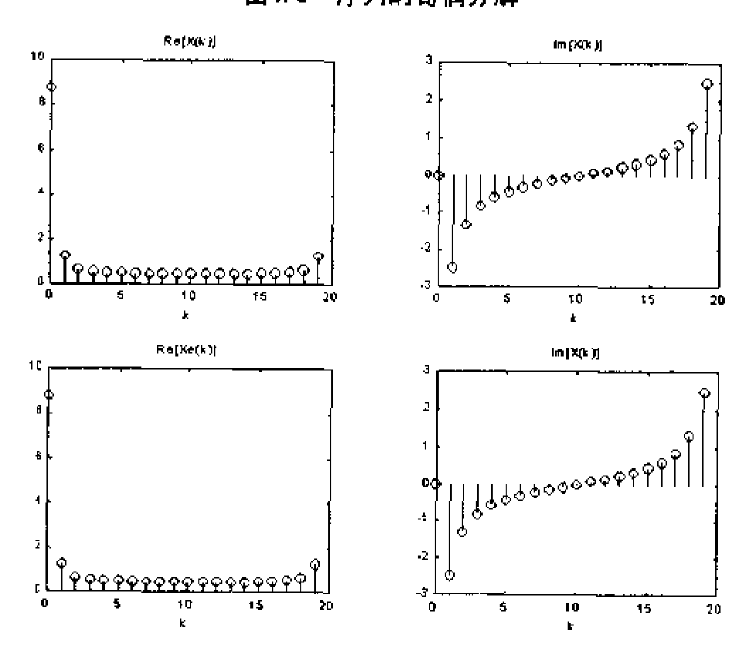


图4.6 X(k)与$X_e(k)$和$X_o(k)$的关系

4.1.4 离散傅立叶变换的快速算法FFT

N 点序列 $x(n)$ 的 DFT 为：

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk} \quad 0\leqslant k\leqslant N-1 \tag{4.1.25}$$

由于系数 $W_N^{nk}=e^{-j\frac{2\pi}{N}nk}$ 是一个周期函数：

$$W_N^{n(N-k)}=W_N^{k(N-n)}=W_N^{-nk} \tag{4.1.26}$$

且是对称的：

$$W_N^{nk+N/2}=-W_N^{nk} \tag{4.1.27}$$

快速傅立叶变换算法正是基于这样的基本思想而发展起来的。它的算法形式有很多种，但基本上可以分成两大类：时间抽取法(DIT-FFT)和频率抽取(DIF-FFT)。由于 DIT-FFT 和 DIF-FFT 算法思想基本一致，只是划分方式略有差异，所以这里以 DIT-FFT 算法为例进行说明。

当 N 是 2 的整数次方时，称为基 2 的 FFT 算法。

首先将序列 $x(n)$ 分解为两组，偶数项为一组，奇数项为一组：

$$\begin{cases}x(2r)=x_1(r)\\x(2r+1)=x_2(r)\end{cases} \quad r=0,1,\cdots,N/2-1 \tag{4.1.28}$$

将 $x_1(r)$ 和 $x_2(r)$ 分别进行 $N/2$ 点的 DFT 得 $X_1(k)$ 和 $X_2(k)$，且：

$$\begin{cases}X(k)=X_1(k)+W_N^kX_2(k)\\X(N/2+k)=X_1(k)-W_N^kX_2(k)\end{cases} \quad k=0,1,\cdots,N/2-1 \tag{4.1.29}$$

重复这一过程，可得到 $x(n)$ 的 FFT。

在 MATLAB 中，可直接利用内部函数 fft 进行计算，它是 MATLAB 系统本身提供的，所以速度比较快。

fft 函数的常用格式为：

```
y=fft(x,n)
```

它用来计算 x 的 n 点 FFT。当 x 的长度小于 n 时，fft 函数在 x 的尾部补零，以构成 n 点数据；当 x 的长度大于 n 时，fft 函数对序列 x 进行截尾。为了提高运算速度，n 通常取 2 的幂次方。

ifft 函数用来计算序列的逆傅立叶变换。

其格式为：

```
y=ifft(x,n)
```

下一节将通过具体的例子来说明 fft 函数的用法。

4.1.5 与 DFT 有关的几个问题

1. 频率分辨率与DFT参数的选择

在讨论 DFT 问题时，频率分辨率是指在频率轴上所能得到的最小频率间隔 Δf 。

我们知道：

$$\Delta f = \frac{f_s}{N}$$

即最小频率间隔Δf反比于数据的长度N。但需要注意的是，这里的数据的长度N必须是数据的有效长度。

如果在 $x(n)$中有两个频率分别为 f_1,f_2 的信号，对 $x(n)$ 用矩形窗截断时，要分辨出这两个频率，N 必须满足：

$$\frac{2f_s}{N} \triangleleft |f_1 - f_2| \quad (4.1.30)$$

补零并没有增加序列的有效长度，所以并不能提高分辨率。但补零可使数据 N 为 2 的整数幂，以便于使用快速离散傅立叶变换算法。补零对原 $X(k)$起到插值的作用，一方面克服“栏栅”效应，平滑谱的外观；另一方面，由于数据截短时引起的频域泄露，有可能在频谱中出现一些难以确认的谱峰，补零后有可能消除这种现象。

下面通过一个例子说明上述两种问题，并阐述 MATLAB 中 fft 函数的用法。

设一序列中含有两种频率成分，f_1= 2Hz，f_2= 2.05Hz，采样频率取为f_s= 10Hz，即

$$x(n) = \sin(2\pi f_1 n / f_s) + \sin(2\pi f_2 n / f_s)$$

根据公式(4.1.30)，要区分出这两种频率成分，必须满足 $N > 400$ 。

(1) 取 $x(n)$ ($0 \leqslant n < 128$)时，计算 $x(n)$ 的 DFT $X(k)$；

(2) 将(1)中的 $x(n)$ 以补零方式使其加长到 $0 \leqslant n < 512$，计算 $X(k)$；

(3) 取 $x(n)$ ($0 \leqslant n < 512$)，计算 $X(k)$ 。

MATLAB 的实现程序如下：

```
%example figure 4.7
function ex47
%
%part 1
%compute X(k) of x(n) 0<=n<256
N=128;
fs=10;%sample frequency unit in Hz
n=0:N-1;
%f1 and f2 are two frequency components of x(n) unit in Hz
f1=2;
f2=2.05;
%generate sequence x(n)
xn=sin(2*pi*f1*n/fs)+sin(2*pi*f2*n/fs);
%compute X(k) using fft function
Xk=fft(xn);
```

```
mXk=abs(Xk(1:N/2));
%plot x(n) and abs(X(k))
figure(1)
subplot(211),plot(n,xn);
xlabel('n');
title('x(n) 0<=n<128');
axis([0 128 -2 2])
k=(0:N/2-1)*fs/N;
subplot(212),plot(k,mXk);
xlabel('frequency in Hz units');
title('magnitude of X(k)');
%
%part 2
%compute X(k) after patching 0
%patch 0
M=512;
xn=[xn zeros(1,M-N)];
%compute X(k) using fft function
Xk=fft(xn);
mXk=abs(Xk(1:M/2));
%plot x(n) and abs(X(k))
figure(2)
n=0:M-1;
subplot(211),plot(n,xn);
xlabel('n');
title('x(n) 0<=n<512');
axis([0 512 -2 2])
k=(0:M/2-1)*fs/M;
subplot(212),plot(k,mXk);
xlabel('frequency in Hz units');
title('magnitude of X(k)');
%
%part 3
%%compute X(k) of x(n) 0<=n<1024
n=0:M-1;
xn=sin(2*pi*f1*n/fs)+sin(2*pi*f2*n/fs);
%compute X(k) using fft function
Xk=fft(xn);
mXk=abs(Xk(1:M/2));
%plot x(n) and abs(X(k))
figure(3)
subplot(211),plot(n,xn);
xlabel('n');
title('x(n) 0<=n<512');
axis([0 512 -2 2])
k=(0:M/2-1)*fs/M;
subplot(212),plot(k,mXk);
xlabel('frequency in Hz units');
title('magnitude of X(k)');
```

执行后得到如图 4.7～4.9 所示的图形。图 4.7 为序列 $x(n)$ $(0\leqslant n<128)$和其 DFT $X(k)$，由于取样点数不满足公式(4.1.30)的要求，所以从图中无法区分出序列中的两种频率成分。图 4.8 为将序列 $x(n)$以补零方式加长到 $0\leqslant n<512$ 后所得到的序列及其 $X(k)$. 从图中可以明显看出，补零对分辨率没有影响，只是对频谱图起到了平滑作用。图 4.9 为采样 512 点的序列 $x(n)$及其 $X(k)$，由于满足了公式(4.1.30)的要求，所以序列中的两个频率成分可以明显区分出来。

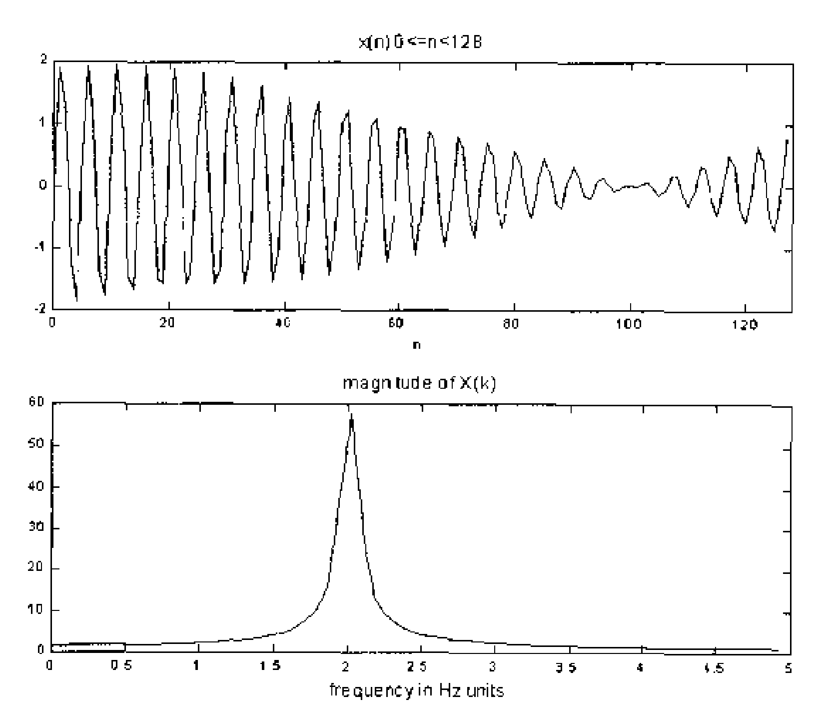


图4.7 序列$x(n)$ $(0\leqslant n<128)$及其DFT $X(k)$

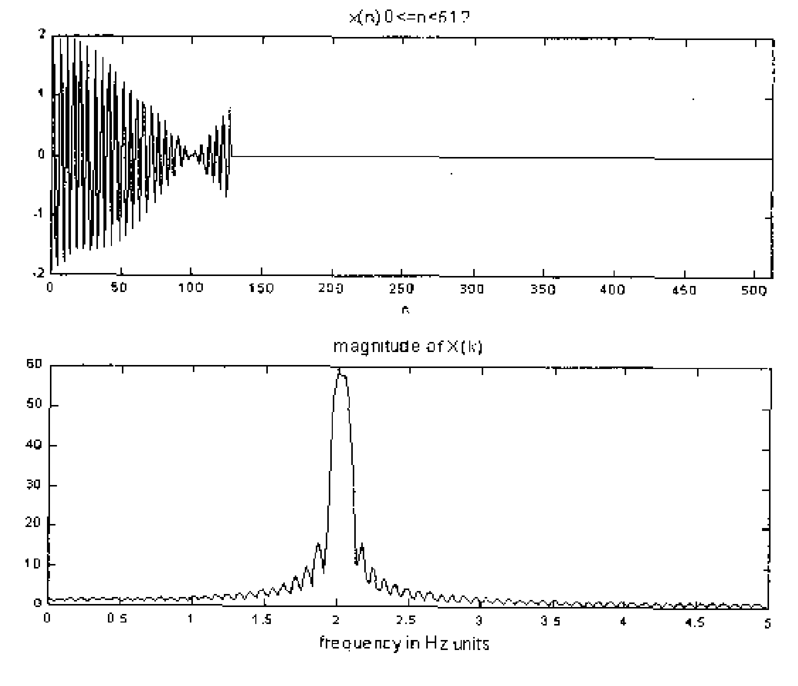


图4.8 补零后的序列及其$X(k)$

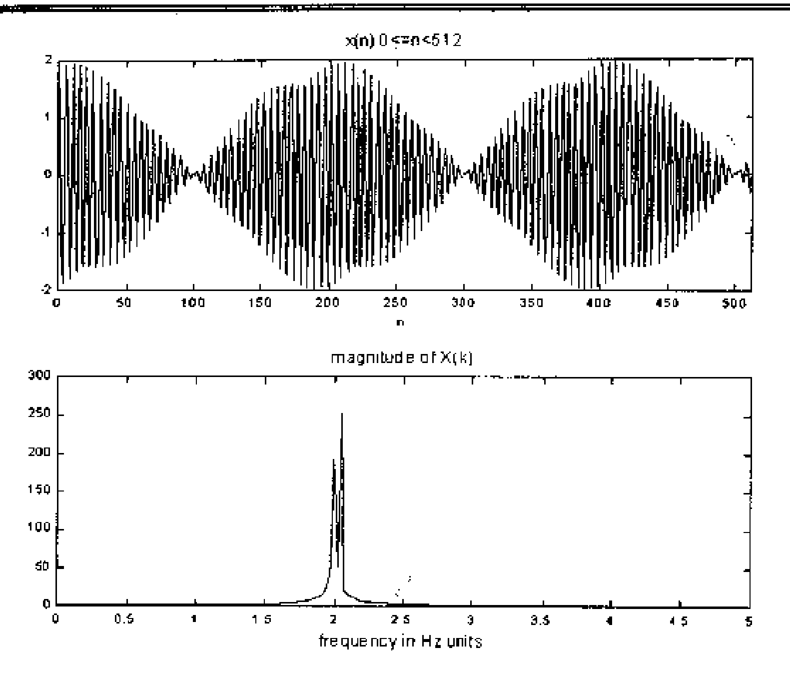


图4.9　序列 $x(n)$ $(0 \leqslant n < 512)$ 及其 $X(k)$

4.2 Z 变换

Z 变换是离散系统与离散信号分析与综合的重要工具，本节主要介绍了有关 Z 变换的定义与收敛域、Z 反变换及 Z 变换的特性等内容，最后给出了用 Z 变换求解差分方程的方法。

4.2.1 Z 变换及其收敛域

一个离散序列 $x(n)$ 的 Z 变换定义为：

$$X(z)=\sum_{n=-\infty}^{\infty} x(n) z^{-n} \tag{4.2.1}$$

这种变换称为双边 Z 变换。当需要考虑序列的起始条件时，常用单边 Z 变换，即：

$$X(z)=\sum_{n=0}^{\infty} x(n) z^{-n} \tag{4.2.2}$$

一般来说，序列的 Z 变换并不一定对任何 z 值都收敛，Z 平面上满足级数收敛的区域称为 Z 变换的收敛域(ROC)，一般形式可表达为：

$$R_{x-}<|z|<R_{x+} \tag{4.2.3}$$

Z 平面上收敛域的位置，或者说 R_{x-} 及 R_{x+} 的大小是和序列有着密切的关系，主要有以下几种情况。

- 有限长序列

序列 $x(n)$ 只在有限的长度 $n_1 \leqslant n \leqslant n_2$ 之内有值，在此长度以外皆为零，即：

$$x(n)=\begin{cases}x(n) & n_1\leqslant n\leqslant n_2\\0 & 其他\end{cases}$$

有限长序列的 Z 变换为：

$$X(z)=\sum_{n=n_1}^{n_2}x(n)z^{-n}$$

其 ROC 为 $|z|>0$

- 右边序列

右边序列为：

$$x(n)=\begin{cases}x(n) & n_1\leqslant n\\0 & 其他\end{cases}$$

右边序列的 Z 变换为：

$$X(z)=\sum_{n=n_1}^{\infty}x(n)z^{-n}$$

其 ROC 为 $|z|>R_{x-}$。

- 左边序列

左边序列为：

$$x(n)=\begin{cases}x(n) & n\leqslant n_2\\0 & 其他\end{cases}$$

左边序列的 Z 变换为：

$$X(z)=\sum_{n=-\infty}^{n_2}x(n)z^{-n}$$

其 ROC 为 $|z|<R_{x+}$。

- 双边序列

一个双边序列可以看作一个左边序列与一个右边序列之和，因此序列的 Z 变换为：

$$X(z)=\sum_{n=-\infty}^{\infty}x(n)z^{-n}$$

$$=\sum_{n=-\infty}^{n_1}x(n)z^{-n}+\sum_{n=n_1+1}^{\infty}x(n)z^{-n}$$

其 ROC 就是这两个序列 Z 变换的公共收敛区间 $R_{x-}<|z|<R_{x+}$。

当 ROC 包含单位圆时，则可计算单位圆上的 Z 变换，实际上它就是离散傅立叶变换：

$$X(z)|_{z=e^{jw}}=X(e^{jw})$$

4.2.2 Z 反变换

Z 反变换的定义为：

$$x(n)=\frac{1}{2\pi \mathrm{j}}\oint_{c} X(z)z^{n-1}\mathrm{d}z \qquad c\in(R_{x-},R_{x+}) \tag{4.2.4}$$

反变换是一个对$X(z)z^{n-1}$进行的围线积分，直接计算围线积分是比较麻烦的，实际上求Z反变换时，往往可以不必直接计算围线积分，一般求解反变换有3种方法：长除法、用留数定律解、部分分式展开法。因为序列的Z变换常为有理函数，因此部分分式展开法比较切合实际，它是利用常用序列的Z变换与留数定律相结合的一种方法。表4.1给出了常用序列的Z变换。

一个因果序列的N阶Z函数，一般可以用N阶的降幂的分子分母多项式表示：

$$X(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_Nz^{-N}}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_Nz^{-N}} \tag{4.2.5}$$

如果$X(z)$的N个极点$\{p_i\}$都是单阶的，那么可以展开成部分分式：

$$X(z)=A_0+\sum_{i=1}^{N}\frac{A_i}{1-p_iz^{-1}} \qquad |z|>\max[|p_i|] \tag{4.2.6}$$

其中A_i是常数，$i=0,1,\cdots,N$。

利用常用序列的Z变换，可以求出式(4.2.6)的反变换为：

表4.1 常见序列的Z变换

序　列	Z变换	收敛域
$\delta(n)$	1	$0\leqslant z\leqslant\infty$
$u(n)$	$\frac{1}{1-z^{-1}}$	$\|z\|>1$
$R_N(n)$	$\frac{1-z^{-N}}{1-z^{-1}}$	$\|z\|>0$
$(n+1)u(n)$	$\frac{1}{(1-z^{-1})^2}$	$\|z\|>1$
$a^nu(n)$	$\frac{1}{1-az^{-1}}$	$\|z\|>a$
$(n+1)a^nu(n)$	$\frac{1}{(1-az^{-1})^2}$	$\|z\|>a$
$e^{jnw_0}u(n)$	$\frac{1}{1-e^{jw_0}z^{-1}}$	$\|z\|>1$
$\sin(nw_0)u(n)$	$\frac{z^{-1}\sin w_0}{1-z^{-1}2\cos w_0+z^{-2}}$	$\|z\|>1$
$\cos(nw_0)u(n)$	$\frac{1-z^{-1}\cos w_0}{1-z^{-1}2\cos w_0+z^{-2}}$	$\|z\|>1$
$e^{-an}\sin(nw_0)u(n)$	$\frac{z^{-1}e^{-a}\sin w_0}{1-z^{-1}2e^{-a}\cos w_0+z^{-2}e^{-2a}}$	$\|z\|>e^{-a}$
$e^{-an}\cos(nw_0)u(n)$	$\frac{1-z^{-1}e^{-a}\cos w_0}{1-z^{-1}2e^{-a}\cos w_0+z^{-2}e^{-2a}}$	$\|z\|>e^{-a}$

$$x(n)=A_0\delta(n)+\sum_{i=1}^{N}A_i p_i^n u(n) \tag{4.2.7}$$

其中常数 A_i 可以利用留数定律求得，将式(4.2.6)改为：

$$\frac{X(z)}{z}=\frac{A_0}{z}+\sum_{i=1}^{N}\frac{A_i}{z-p_i}$$

可见 A_i 分别是 $X(z)/z$ 在极点 0，p_i 处的留数，因此：

$$A_0=\mathrm{Res}\left[\frac{X(z)}{z},0\right]=X(0)=\frac{b_N}{a_N}$$

$$A_i=\mathrm{Res}\left[\frac{X(z)}{z},p_i\right]=(1-p_i z^{-1})X(z)|_{z=p_i} \tag{4.2.8}$$

在 MATLAB 中，函数 residuez 可以实现上述过程，其格式为：

```
[R,P,K] = RESIDUEZ(B,A)
```

参数 B、A 分别为有理 Z 函数的分子多项式的系数向量与分母多项式的系数向量，参数 R 与 P 为列向量，R 为留数，P 为极点，如果分子多项式的阶数大于分母多项式的阶数，参数 K 返回直接项系数。

极点数为：

```
n = length(A)-1 = length(R) = length(P)
```

如果 length(B) < length(A)，则直接项系数 K 为空，否则

```
length(K) = length(B)-length(A)+1
```

例如，计算

$$X(z)=\frac{1}{(1+0.2z^{-1})(1-0.2z^{-1})(1+0.6z^{-1})(1-0.6z^{-1})} \qquad |z|>0.6$$

的 Z 反变换。

MATLAB 实现程序如下：

```
b=1;
a=poly([-0.2 0.2 -0.6 0.6]);
[R,P,K]=residuez(b,a)
```

执行结果为：

```
R =
    0.5625
    0.5625
   -0.0625
   -0.0625
P =
    0.6000
   -0.6000
    0.2000
   -0.2000
K =
```

[]

因此得到：

$$X(z)=\frac{0.5625}{1-0.6z^{-1}}+\frac{0.5625}{1+0.6z^{-1}}-\frac{0.0625}{1-0.2z^{-1}}-\frac{0.0625}{1+0.2z^{-1}}$$

相应的 Z 反变换为：

$$x(n)=[0.5625(0.6)^n+0.5625(-0.6)^n-0.0625(0.2)^n-0.0625(-0.2)^n]u(n)$$

4.2.3　Z 变换的特性

1.　线性

若

$$Z[x(n)]=X(z)\qquad R_{x-}<|z|<R_{x+}$$
$$Z[y(n)]=Y(z)\qquad R_{y-}<|z|<R_{y+}$$

则

$$Z[ax(n)+by(n)]=aX(z)+bY(z)\qquad R_-<|z|<R_+$$

其中 $R_-=\max[R_{x-},R_{y-}]$　　$R_+=\min[R_{x+},R_{y+}]$

2.　序列移位

$$Z[x(n-n_0)]=z^{-n_0}X(z)\qquad R_{x-}<|z|<R_{x+}$$

3.　与指数序列相乘

$$Z[a^n x(n)]=X(a^{-1}z)\qquad |a|R_{x-}<|z|<|a|R_{x+}$$

4.　$X(z)$ 的微分

$$Z[nx(n)]=-z\frac{dX(z)}{dz}\qquad R_{x-}<|z|<R_{x+}$$

5.　复序列的共轭

$$Z[x^*(n)]=X^*(z^*)\qquad R_{x-}<|z|<R_{x+}$$

6.　序列卷积

$$Z[x(n)*y(n)]=X(z)Y(z)\qquad R_-<|z|<R_+$$

其中 $R_-=\max[R_{x-},R_{y-}]$　　$R_+=\min[R_{x+},R_{y+}]$

7.　序列乘积

$$Z[x(n)y(n)]=\frac{1}{2\pi j}\oint_c X(v)Y(\frac{z}{v})v^{-1}dv\qquad R_{x-}R_{y-}<|z|<R_{x+}R_{y+}$$

例如，序列 $x(n)$、$y(n)$ 分别为：

$$x(n)=3\delta(n+2)+2\delta(n+1)+4\delta(n)+\delta(n-1)$$

$$y(n) = 4\delta(n) + 5\delta(n-1) + 3\delta(n-2) + 2\delta(n-3)$$

其相应的 Z 变换分别为：

$$X(z) = 3z^2 + 2z + 4 + z^{-1}$$

$$Y(z) = 4 + 5z^{-1} + 3z^{-2} + 2z^{-3}$$

试求 $X(z)Y(z)$。

MATLAB 的实现程序为：

```
x=[3 2 4 1];
n1=-2:1;
y=[4 5 3 2];
n2=0:3;
ns=n1(1)+n2(1);
ne=n1(length(x))+n2(length(y));
n=ns:ne
z=conv(x,y)
```

执行后得到结果为：

```
n =
    -2    -1     0     1     2     3     4
z =
    12    23    35    36    21    11     2
```

则

$$X(z)Y(z) = 12z^2 + 23z + 35 + 36z^{-1} + 21z^{-2} + 11z^{-3} + 2z^{-4}$$

4.2.4 用 Z 变换求解差分方程

一个 LSI 系统，其差分方程可表示为：

$$y(n) = -\sum_{k=1}^{N} a(k)y(n-k) + \sum_{r=0}^{M} b(r)x(n-r) \tag{4.2.9}$$

给定 $x(n)$ 及 $y(n)$ 的初始条件，我们希望得到序列 $y(n)$ 的闭合表达式，这即是差分方程的求解问题。

对于齐次方程，即 $x(n)=0$，

$$y(n) + \sum_{k=1}^{N} a(k)y(n-k) = 0 \tag{4.2.10}$$

方程的解是由 $y(n)$ 的初始条件引起的，称为零输入解。对该式两边取 Z 变换，并令 $a(0)=1$，则

$$Y(z) = \frac{-\sum_{k=0}^{N} a(k)z^{-k}[\sum_{m=-k}^{-1} y(m)z^{-m}]}{\sum_{k=0}^{N} a(k)z^{-k}} \tag{4.2.11}$$

取 Z 反变换，即得系统的零输入解：

$$y_{0i}(n)=\mathrm{Z}^{-1}[Y(z)]$$

若 $y(n)$ 的初始条件等于零，且 $x(n)$ 是因果序列，那么式(4.2.9)的 Z 变换为：

$$Y(z)=\frac{\sum_{r=0}^{M} b(r)z^{-r}}{1+\sum_{k=0}^{N} a(k)z^{-k}}=H(z)X(z) \tag{4.2.12}$$

由此得到的 $y(n)$ 称为零状态解，它是单纯由输入所引起的输出，即

$$y_{0s}(n)=\mathrm{Z}^{-1}[H(z)X(z)]$$

系统完整的输出应是零状态解与零输入解之和，即

$$y(n)=y_{0i}(n)+y_{0s}(n)$$

例如，令 $y(n)-2y(n-1)+3y(n-2)=u(n)+ux(n-1)+5u(n-2)-6u(n-3)$，初始条件为 $x(-1)=1$，$x(-2)=1$，$y(-1)=-1$，$y(-2)=1$。

求 $y(n)$。

MATLAB 实现程序为：

```
b=[1 4 5 -6];
a=[1 -2 3];
x0=[1 1];
y0=[-1 1];
xic=filtic(b,a,y0,x0)
bxplus=1;
axplus=[1 -1];
ayplus=conv(a,axplus)
byplus=conv(b,bxplus)+conv(xic,axplus)
[R,P,K]=residuez(byplus,ayplus)
Mp=abs(P)
Ap=angle(P)*180/pi
N=100;
n=0:N-1;
xn=ones(1,N);
yn=filter(b,a,xn,xic);
plot(n,yn)
```

执行后得到的结果为：

```
xic =
     4     2    -6
ayplus =
     1    -3     5    -3
byplus =
     5     2    -3     0
R =
    1.5000 - 4.2426i
```

```
      1.5000 + 4.2426i
      2.0000
  P =
      1.0000 + 1.4142i
      1.0000 - 1.4142i
      1.0000
  K =
      [ ]
  Mp =
     1.7321
     1.7321
     1.0000
  Ap =
      54.7356
     -54.7356
          0
```

其单位阶跃响应曲线如图 4.10 所示。

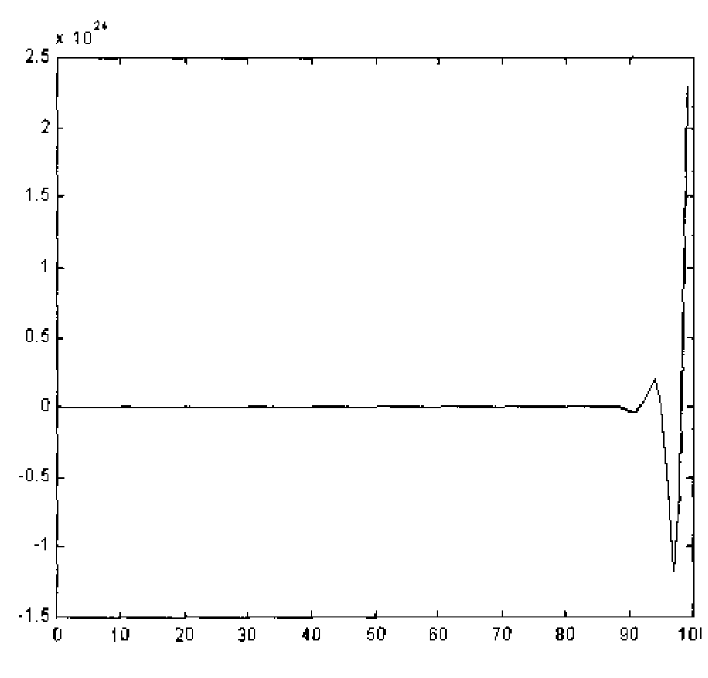


图4.10 系统的单位阶跃响应

4.3 Chirp Z 变换

Chirp Z 变换(CZT)即线性调频 Z 变换，可用来计算单位圆上任一段曲线上的 Z 变换，做 DFT 时输入的点数 N 和输出点数 M 可以不相等，从而达到频域“细化”的目的。

4.3.1 Chirp Z 变换的定义

序列 $x(n)$ 的 Z 变换

$$X(z)=\sum_{n=0}^{\infty}x(n)z^{-n}$$

令 $z_r=AW^{-r}$

式中 $A=A_0e^{j\theta_0}$ $W=W_0e^{j\varphi_0}$

则
$$z_r=A_0e^{j\theta_0}W_0^{-r}e^{j\varphi_0 r} \tag{4.3.1}$$

A_0,W_0为任意的正实数，给定$A_0,W_0,\theta_0,\varphi_0$，当$r=0,1,\cdots,\infty$时，可得到在z平面上的一系列点$z_0,z_1,\cdots,z_\infty$，取这些点的Z变换，有

$$X(z_r)=\sum_{n=0}^{\infty}x(n)z_r^{-n}=\sum_{n=0}^{\infty}x(n)A^{-n}W^{nr} \tag{4.3.2}$$

这即是CZT的定义。

当$r=0$时，$z_0=A_0e^{j\theta_0}$，该点在z平面上的幅度为A_0，幅角为θ_0，是CZT的起点。随着r的变化，点$z_0,z_1,\cdots,z_\infty$构成了CZT变换的路径，它是一条螺旋线，且具有下述特点。

(1) 当$A_0>1$时，螺旋线在单位圆外，反之，在单位圆内。

(2) 当$W_0>1$时，$A_0W_0^{-1}<A_0$，螺旋线内旋，反之，螺旋线外旋。

(3) 当$A_0=W_0=1$时，CZT的变换路径为单位圆上的一段圆弧。

(4) 当$A_0=W_0=1$，$\theta_0=0$，$M=N$时，CZT变成普通的Z变换。

因为我们希望得到的是信号的频谱分析，故应在单位圆上去实现CZT，即应取$A_0=W_0=1$。

对于长度为N的序列$x(n)$，其在单位圆上的M点CZT变换为

$$X(z_r)=\sum_{n=0}^{N-1}x(n)A^{-n}W^{nr}\quad r=0,1,\cdots,M-1 \tag{4.3.3}$$

由于
$$nr=\frac{1}{2}[r^2+n^2-(r-n)^2]$$

所以

$$X(z_r)=\sum_{n=0}^{N-1}x(n)A^{-n}W^{r^2/2}W^{n^2/2}W^{-(r-n)^2/2} \tag{4.3.4}$$

令
$$g(n)=x(n)A^{-n}W^{n^2/2} \tag{4.3.5}$$

$$h(n)=W^{-n^2/2} \tag{4.3.6}$$

则
$$X(z_r)=W^{r^2/2}\sum_{n=0}^{N-1}g(n)h(r-n)$$

$$=W^{r^2/2}[g(r)*h(r)]=W^{r^2/2}y(r) \tag{4.3.7}$$

式中
$$y(r)=g(r)*h(r)=\sum_{n=0}^{N-1}g(n)W^{-\frac{(r-n)^2}{2}} \tag{4.3.8}$$

4.3.2 Chirp Z变换的计算方法

由上面的理论可知，计算单位圆上CZT的关键是实现$g(n)$和$h(n)$的线性卷积，这可以

通过快速傅立叶变换来实现，但需对序列 $g(n)$ 和 $h(n)$ 进行预处理。具体处理方法以及 CZT 的计算步骤如下：

(1) 按式(4.3.5)计算出 $g(n)$，$n=0,1,\cdots,N-1$，并将 $g(n)$ 补零，使之长度为 L，得到新序列 $g'(n)$

$$g'(n)=\begin{cases}g(n) & n=0,1,\cdots,N-1\\0 & n=N,N+1,\cdots,L\end{cases} \tag{4.3.9}$$

(2) 将 $h(n)$ 也转换成一个 L 点的新序列 $h'(n)$

$$h'(n)=\begin{cases}h(n) & n=0,1,\cdots,M-1\\0 & n=M,M+1,\cdots,L-N\\h(L-n) & L-N+1\leqslant n\leqslant L-1\end{cases} \tag{4.3.10}$$

上面两式中 L 的选择应满足 $L\geqslant N+M-1$，且取 2 的整数次幂。

(1) 求序列 $g'(n)$ 和 $h'(n)$ 的 DFT $G'(k)$ 与 $H'(k)$，它们都是 L 点序列。

(2) 令 $Y'(k)=G'(k)H'(k)$，并求其反变换，得 $y(r)$，仅取 $y(r)$ 的前 M 个点。

(3) 用 $W^{r^2/2}$ 乘 $y(r)$，则得最后的输出 $X(z_r)$。

4.3.3 Chirp Z 变换的 MATLAB 实现

在 MATLAB 中实现线性调频 Z 变换很简单，只需调用工具箱中的 czt 函数即可。

czt 函数的调用格式为：

```
y=czt(x,m,w,a]
```

它用来计算序列 x 沿着由 w 和 a 定义的螺旋线上的 Z 变换。m 指定变换长度，w 指定沿着 z 平面螺旋线上的点之间的比率，a 指定起始点。当 m,w,a 未指定时，其相当于 FFT。

下面通过例子来说明 czt 函数的用法。

例，已知 序列 $x(n)=(0.9)^n(0\leqslant n<10)$

(1) 当 $\theta_0=\frac{\pi}{4}$，$\varphi_0=\frac{\pi}{6}$，$M=6$ 时，求序列 $x(n)$ 在单位圆上的 CZT。

(2) 比较 czt(x)与 fft(x)的结果。

MATLAB 程序为：

```
%example figure 4.11
function ex411
N=10;%length of sequence x(n)
n=0:N-1;
xn=(0.9).^n;
%
%part 1 compute Chirp z transform of sequence x(n)
sita=pi/4;%phase of start point
fai=pi/6;
A=exp(j*sita);%complex starting point
W=exp(-j*fai);%complex ratio between points on the contour
M=6;%length of the transform
```

```
y=czt(xn,M,W,A);
%plot x(n) and y
subplot(211),stem(r,xn);
xlabel('n');title('sequence x(n)');
r=0:M-1;
subplot(212),stem(r,abs(y));
xlabel('r');title('CZT of x(n)');
%
%part 2 compare fft(xn) and czt(xn)
Xk=fft(xn)
CXk=czt(xn)
```

执行后输出结果：

```
Xk=
    6.5132    0.5006 - 0.9739i    0.3750 - 0.4447i    0.3518 - 0.2356i
    0.3446 - 0.1055i    0.3428 - 0.0000i    0.3446 + 0.1055i    0.3518 + 0.2356i
    0.3750 + 0.4447i    0.5006 + 0.9739i
CXk=
    6.5132 - 0.0000i    0.5006 - 0.9739i    0.3750 - 0.4447i    0.3518 - 0.2356i
    0.3446 - 0.1055i    0.3428 - 0.0000i    0.3446 + 0.1055i    0.3518 + 0.2356i
    0.3750 + 0.4447i    0.5006 + 0.9739i
```

得到如图 4.11 所示的图形。

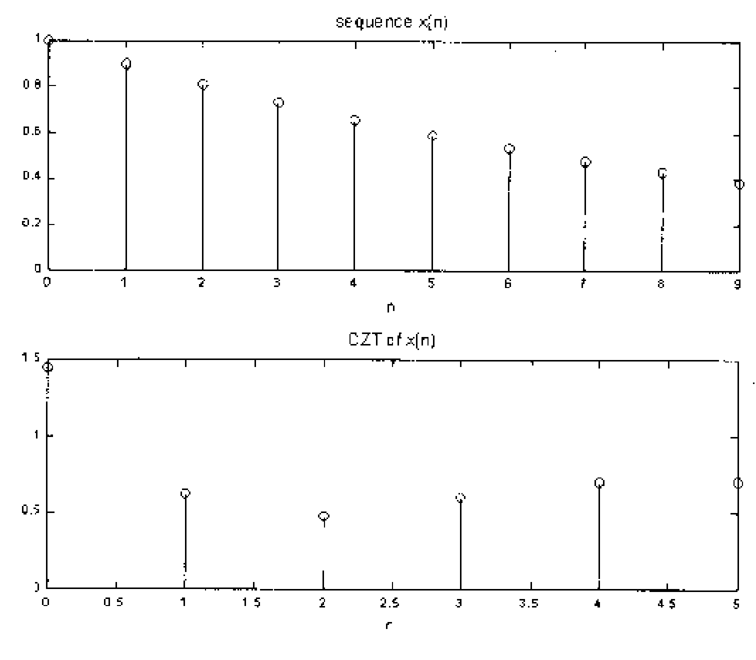


图4.11 序列$x(n)$及其CZT

4.4 离散余弦变换

本节首先给出了离散余弦变换的定义，接着介绍了离散余弦变换的 MATLAB 实现方法。

4.4.1 离散余弦变换(DCT)的定义

N 点序列 $x(n)$ 的离散余弦变换定义为：

$$\begin{cases} X_c(0) = \dfrac{1}{\sqrt{N}}\displaystyle\sum_{n=0}^{N-1} x(n) \\ X_c(k) = \sqrt{\dfrac{2}{N}}\displaystyle\sum_{n=0}^{N-1} x(n)\cos\dfrac{(2n+1)k\pi}{2N} \quad k = 1,2,\cdots,N-1 \end{cases} \tag{4.4.1}$$

显然，其变换的核函数为：

$$C_{k,n} = \sqrt{\frac{2}{N}} g_k \cos\frac{(2n+1)k\pi}{2N} \tag{4.4.2}$$

式中系数

$$g_k = \begin{cases} 1/\sqrt{2} & k = 0 \\ 1 & k \neq 0 \end{cases} \tag{4.4.3}$$

这样，如果序列 $x(n)$ 是实数，那么它的 DCT 也是实数。然而对离散傅立叶变换来说，即使序列 $x(n)$ 是实数，其 DFT 一般为复数，由此可以看出，离散余弦变换避免了复数运算。

用矩阵形式表示的离散余弦变换为：

$$X_c = C_N x \tag{4.4.4}$$

式中 X_c，x 都是 $N \times 1$ 的向量，分别代表输出的离散余弦变换序列和输入序列，C_N 是 $N \times N$ 的变换矩阵，元素可以根据(4.3.2)式求出。例如当 $N = 4$ 时，有

$$C_4 = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{\sqrt{2}}{2}\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ \cos\frac{\pi}{8} & \cos\frac{3\pi}{8} & \cos\frac{5\pi}{8} & \cos\frac{7\pi}{8} \\ \cos\frac{2\pi}{8} & \cos\frac{6\pi}{8} & \cos\frac{10\pi}{8} & \cos\frac{14\pi}{8} \\ \cos\frac{3\pi}{8} & \cos\frac{9\pi}{8} & \cos\frac{15\pi}{8} & \cos\frac{21\pi}{8} \end{bmatrix} \tag{4.4.5}$$

可以证明，C_N 的行、列向量均有下述关系：

$$\langle c_k, c_n \rangle = \begin{cases} 1 & k = n \\ 0 & k \neq n \end{cases}$$

所以变换矩阵 C_N 是归一化的正交阵，DCT 是正交变换。

DCT 的反变换为：

$$x(n) = \frac{1}{\sqrt{N}} X_c(0) + \sqrt{\frac{2}{N}}\sum_{k=1}^{N-1} X_c(k)\cos\frac{(2n+1)k\pi}{2N}$$
$$n = 0,1,\cdots,N-1 \tag{4.4.6}$$

用矩阵形式表示为：

$$x = C_N^{-1} X_c = C_N^T X_c \tag{4.4.7}$$

4.4.2　离散余弦变换(DCT)的 MATLAB 实现

由于离散余弦变换及其反变换可以用矩阵形式表示，所以用 MATLAB 计算起来非常方便。在 MATLAB 中，dct 函数可以实现一个序列的离散余弦变换，idct 函数可以实现一个序列的离散余弦反变换。

它们的格式非常简单：

```
y=dct(x,n)
x=idct(y,n)
```

其中参数 n 表示在变换前将序列 x 或 y 补足或截短至长度为 n。

由于 DCT 具有很好的能量压缩性，仅用几个变换系数即可代表序列能量的总体，所以其在数据通讯中具有广泛的应用。下面通过一例子进行说明：

例，已知一余弦序列

$$x(n)=\cos(2\pi fn/f_s)\quad 0\leqslant n<1000$$

其中 $f=50$ Hz，f_s=1000Hz。

计算此序列的 DCT 并仅用幅值大于 5 的部分重建信号。

MATLAB 实现程序如下：

```
%example figure 4.12
function ex412
f=50;%sequence frequency
fs=1000;%sample frequency
N=1000;
n=0:N-1;
xn=cos(2*pi*f*n/fs);
%part 1 compute DCT of x(n)
y=dct(xn);
%part 2 set values of abs(y)<5 0
num=find(abs(y)<5);
y(num)=zeros(size(num));
%part 3 compute IDCT
zn=idct(y);
%part 4 compare z(n) and x(n)
figure(2)
subplot(211),plot(n,xn)
xlabel('n');
title('x(n)');
subplot(212),plot(n,zn)
xlabel('n');
title('z(n)');
rp=100-norm(xn-zn)/norm(xn)*100
```

执行上述程序，得到的结果为：

```
rp =
    84.3566
```

也就是说，信号重建率为 84.3566%。原始序列 $x(n)$ 与重建序列 $z(n)$ 的图形如图 4.12 所示。

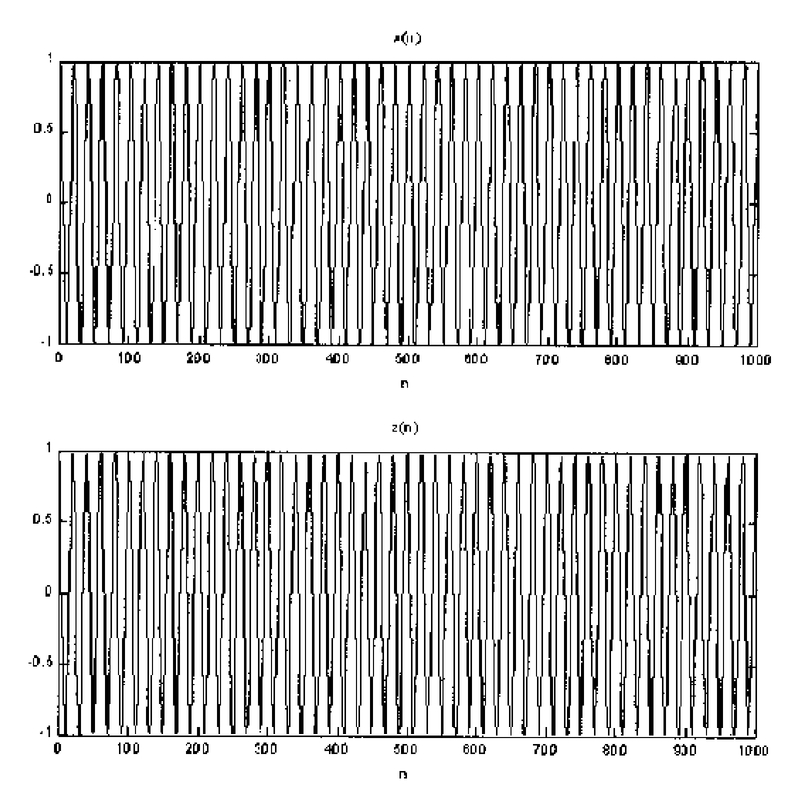


图4.12 原始序列$x(n)$与重建序列$z(n)$

4.5 Hilbert 变换

4.5.1 Hilbert 变换的定义

Hilbert 变换器的单位抽样响应 $h(n)$ 为：

$$h(n)=\frac{1-(-1)^n}{n\pi}=\begin{cases}0 & n\text{为偶数}\\ \dfrac{2}{n\pi} & n\text{为奇数}\end{cases} \tag{4.5.1}$$

设序列 $x(n)$ 的 Hilbert 变换是 $\hat{x}(n)$，则

$$\hat{x}(n)=x(n)*h(n)=\frac{2}{\pi}\sum_{m=-\infty}^{\infty}\frac{x(n-2m-1)}{(2m+1)} \tag{4.5.2}$$

求出 $\hat{x}(n)$ 后，即可构成 $x(n)$ 的解析信号：

$$z(n)=x(n)+j\hat{x}(n) \tag{4.5.3}$$

也可用 DFT 方便地求出一个信号 $x(n)$ 的解析信号及 Hilbert 变换，步骤是：

(1) 求 $x(n)$ 的 DFT $X(k)$，$k=0,1,\cdots,N-1$，其中 $k=\frac{N}{2},\cdots,N-1$ 对应负数频率。

(2) 令

$$Z(k)=\begin{cases}X(k) & k=0\\ 2X(k) & k=1,2,\cdots,N/2-1\\ 0 & k=N/2,\cdots,N-1\end{cases} \tag{4.5.4}$$

(3) 对 $Z(k)$ 做逆 DFT，即得到 $x(n)$ 的解析信号 $z(n)$。

(4) 由 $Z(k)=X(k)+j\hat{X}(k)$，得

$$\hat{x}(n)=-j[z(n)-x(n)] \tag{4.5.5}$$

4.5.2 Hilbert 变换的 MATLAB 实现

基于上述利用 DFT 求解一个序列的解析信号及 Hilbert 变换的理论，利用 MATLAB 语言编写出了扩展函数 yhilbert.m，其程序如下：

```
function [zn,hxn]=yhilbert(xn)
%this program is used to compute Hilbert
%transform of the sequence x(n)
%Method: based on DFT
%First:compute X(k)=DFT[x(n)]
%Second:
   %Z(k)=X(k) if k=0
   %Z(k)=2X(k) if 1<=k<=N/2-1
   %Z(k)=0    if N/2<=k<=N-1
%Third:compute analytic sequence z(n)=IDFT[Z(k)];
%Forth:compute Hilbert transform of the sequence x(n)
   % Hilbert[x(n)]=-j[z(n)-x(n)]
%zn is analytic sequence of x(n)
%hxn is Hilbert transform of x(n)
N=length(xn);
Xk=fft(xn);
k1=0;
Zk(k1+1)=Xk(k1+1);
k2=1:N/2-1;
Zk(k2+1)=2*Xk(k2+1);
k3=N/2:N-1;
Zk(k3+1)=zeros(size(k3));
zn=ifft(Zk);
hxn=-j*(zn-xn);
```

另外，MATLAB 工具箱本身也提供了计算 Hilbert 变换的函数 hilbert.m，其格式为 y=Hilbert(x)。但需注意的是，该函数计算出的结果是序列的解析信号，其虚部才是序列的 Hilbert 变换。

4.5.3 Hilbert 变换的性质

Hilbert 变换具有两个性质：

性质 1，序列 $x(n)$通过 Hilbert 变换器后，信号频谱的幅度不发生变化。这是因为 Hilbert

变换器是全通滤波器，引起频谱变化的只是其相位。

性质 2，序列 $x(n)$ 与其 Hilbert 变换 $\hat{x}(n)$ 是正交的。

下面通过一个例子来验证上述两条性质，并对扩展函数 yhilbert.m 与 MATLAB 工具箱中的函数 hilbert.m 进行比较。

例，已知序列

$$x(n)=\cos(0.2\pi n) \quad 0 \leqslant n < 20$$

(1) 计算序列 $x(n)$ 的 Hilbert 变换 $\hat{x}(n)$，并比较两序列频谱的变化。

(2) 验证 $x(n)$ 与 $\hat{x}(n)$ 是正交的。

(3) 余弦序列的 Hilbert 变换是正弦序列，则序列 $x(n)$ 的 Hilbert 变换为 $\hat{x}(n)=\sin(0.2\pi n)$，试比较扩展函数 yhilbert.m 与函数 hilbert.m 对序列 $x(n)$ 进行 Hilbert 变换的结果。

MATLAB 实现程序如下：

```
%example figure 4.13
function ex413
N=20;
n=0:N-1;
xn=cos(0.2*pi*n);
[zn,hxn]=yhilbert(xn);
%part 1
%compare FFT[x(n)] and FFT[Hilbert[x(n)]]
Xk=fft(xn);
hXk=fft(hxn);
aXk=abs(Xk);
ahXk=abs(hXk);
pXk=phase(Xk);
phXk=phase(hXk);
k=0:N-1;
subplot(221),stem(k,aXk)
xlabel('k');
title('amplitude of FFT[x(n)]');
subplot(222),stem(k,pXk)
xlabel('k');
title('phase of FFT[x(n)]');
subplot(223),stem(k,ahXk)
xlabel('k');
title('amplitude of FFT[Hilber[x(n)]]');
subplot(224),stem(k,phXk)
xlabel('k');
title('phase of FFT[Hilber[x(n)]]');
%part 2: verify verticality of x(n) and Hilbert[x(n)]
%Method: if sum(xn.*hxn) is colsely 0
add=sum(xn.*hxn)
%part 3 :compare expansion function yhilbert.m and hilbert.m
y=hilbert(xn);
hxn
sn=sin(0.2*pi*n)
yn=imag(y)
```

执行后得到如下结果:

```
add =
     1.3905e-014
hxn=
    -0.0000  0.5878  0.9511  0.9511 0.5878  0.0000  -0.5878 -0.9511
    -0.9511 -0.5878 -0.0000  0.5878 0.9511  0.9511   0.5878  0.0000
    -0.5878 -0.9511 -0.9511 -0.5878
sn=
       0     0.5878  0.9511  0.9511 0.5878  0.0000  -0.5878 -0.9511
    -0.9511 -0.5878 -0.0000  0.5878 0.9511  0.9511   0.5878  0.0000
    -0.5878 -0.9511 -0.9511 -0.5878
yn=
    -0.3523  0.6368  0.8847  0.9945  0.5628  0.0391 -0.5990
    -0.9133 -0.9556 -0.5482 -0.0000  0.6329  0.9556  1.0084
     0.5990  0.0840 -0.5628 -0.7946 -0.8847 -0.0212
```

得到如图 4.13 所示的图形。

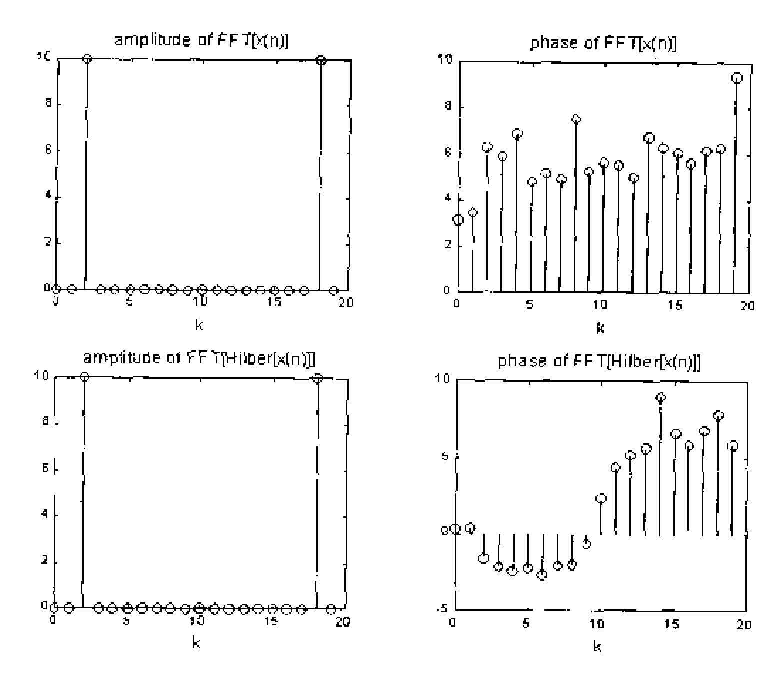


图4.13　Hilbert变换前后频谱变化比较

从结果中的 add 值可以看出，$x(n)$ 与 $\hat{x}(n)$ 乘积和近似为零，即它们是正交的；比较利用扩展函数 yhilbert.m 得到的 Hilbert 变换结果 hxn、利用函数 Hilbert.m 得到的 Hilbert 变换结果 yn 以及理论上的 Hilbert 变换结果 sn，可以看出扩展函数 yhilbert.m 较 MATLAB 工具箱中的函数 hilbert.m 精确。

第 5 章　离散系统的结构及其 MATLAB 实现

AR 模型功率谱估计是现代谱估计的主要内容，本节首先给出了 AR 模型的 Yule-Walker 方程，然后介绍了 Levinson-Durbin 递推算法与 AR 模型参数的其它求解算法，并讨论了有关 AR 模型阶数 p 的选择问题，最后讲述了 MATLAB 中 AR 模型谱估计的函数及 AR 模型谱估计的性质。

5.1　IIR 系统的结构

一个 N 阶 IIR 数字滤波器(DF)的系统函数可表示为：

$$H(z)=\frac{\sum_{k=0}^{M}b_k z^{-k}}{1+\sum_{k=1}^{N}a_k z^{-k}} \tag{5.1.1}$$

其差分方程为：

$$y(n)=\sum_{k=0}^{M}b_k x(n-k)-\sum_{k=0}^{N}a_k y(n-k) \tag{5.1.2}$$

无限长单位取样响应 IIR 数滤波器的主要特点是：

- 单位取样响应是无限长的，即 h(n)，$-\infty<n<\infty$。
- 系统函数 H(z)在有限平面 z 上有极点存在。
- 结构上存在着输出到输入的反馈网络，即结构是递归的。

实现同一个系统函数 H(z)，可以用不同的结构形式，它的主要的结构形式有直接 I 型、直接 II 型、级联型与并联型几种。

5.1.1　直接 I 型

从(5.1.2)式的差分方程可以看出，y(n)由两部分构成。

第 1 部分[$\sum_{k=0}^{M}b_k x(n-k)$]表示将输入信号进行延时，组成 M 节的延时网络，把每节延时抽头后与常系数 b_k 相乘，然后再把结果相加，这是一个横向结构网络，即实现零点的网络。

第 2 部分[$\sum_{k=0}^{N} a_k y(n-k)$]表示将输出信号进行延时，组成 N 节的延时网络，把每节延时抽头后与常系数 a_k 相乘，然后再把结果相加。由于这部分是对输出的延时，故为反馈网络。这部分网络实现极点。

y(n)是由以上这两部分相加组成。其信号流图如图 5.1 所示。

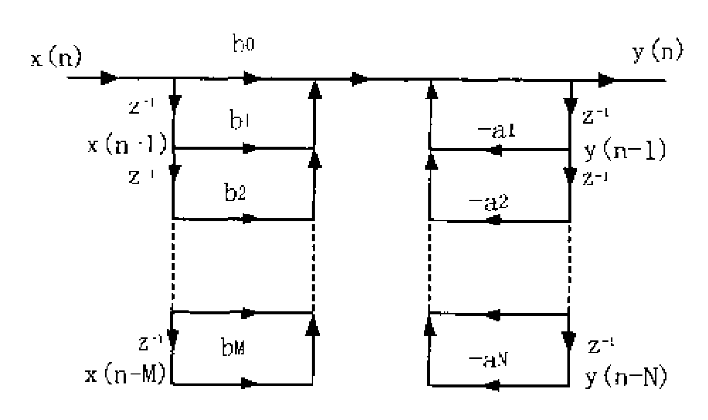


图5.1　直接I型

由于系数组 b_k 相应于 H(z)的分子多项式，而系数组 b_k 相应于 H(z)的分母多项式，因此可以将图 5.1 解释为由两个网络级联组成。在线性非时变系统情况下，级联系统总的输入一输出关系和子系统的次序无关。如果先实现 H(z)的极点，后实现 H(z)的零点，并且合并 z^{-1} 的支路，则可得出直接 II 型结构。

5.1.2　直接 II 型

将(5.1.1)改写为：

$$H(z)=(\sum_{k=0}^{M} b_k z^{-k})[\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}]=H_1(z)\cdot H_2(z) \tag{5.1.3}$$

如图 5.1 所示的直接 I 型结构，可以看成是两部分网络的级联。我们知道一个线性时不变系统，如果交换其子系统的次序，系统函数不变。则对应的信号流图如图 5.2 所示。

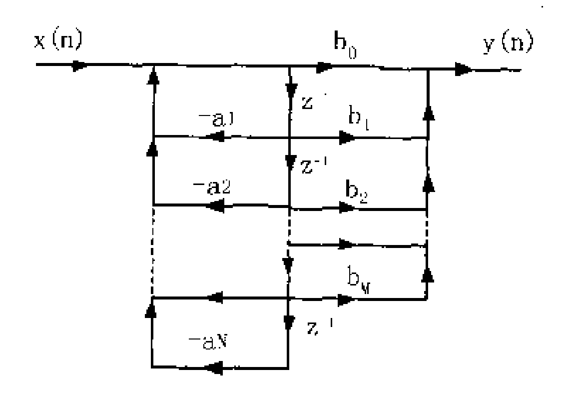


图5.2　直接II型

对于N阶差分方程，直接II型结构只需N个延时单元，比直接I型结构的延时单元少一半。因而在软件实现时可以节省存储单元，而在硬件实现时，可节省寄存器，故比直接I型好。

在MATLAB中，可利用DSP工具函数filter实现IIR的直接形式。Filter函数的用法是：

```
Y=filter(B,A,X);
```

其中，B表示系统转移函数的分子多项式的系数矩阵，A表示系统转移函数的分母多项式的系数矩阵，X表示输入序列，Y即为输出序列。具体用法可以在MATLAB的帮助文件中找到。

例，有一滤波器，

$$16y(n)+12y(n-1)+2y(n-2)-4y(n-3)-y(n-4)=x(n)-3x(n-1)+11x(n-2)-27(n-3)+18x(n-4)$$

输入为单位冲激响应，求输出。

解：

在MATLAB的主窗口中输入如下：

```
%输入系数矩阵
b=[1,-3,11,-27,18];
a=[16,12,2,-4,-1];
%输入单位响应序列
x=[1,zeros(1,100)];
%滤波器输出
y=filter(b,a,x)
```

画出的输出波形如图5.3所示。

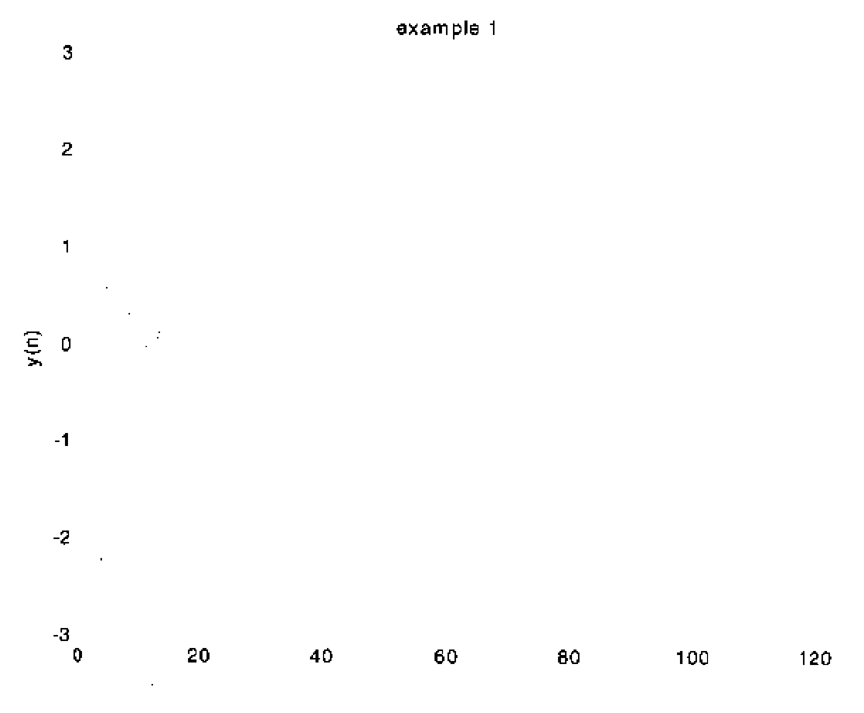


图5.3 系统输出波形

5.1.3 级联型

直接形式网络结构可直接从(5.1.1)式的系统函数得到。如果把分子分母多项式进行因式分解，则可将 H(z)写成：

$$H(z)=A\frac{\prod_{k=1}^{M_1}(1-g_kz^{-1})\prod_{k=1}^{M_2}(1-h_kz^{-1})(1-h_k^*z^{-1})}{\prod_{k=1}^{N_1}(1-c_kz^{-1})\prod_{k=1}^{N_2}(1-d_kz^{-1})(1-d_k^*z^{-1})} \tag{5.1.4}$$

式中的 $M=M_1+2M_2$ 和 $N=N_1+2N_2$。在这个表示式中，一阶因式表示实零点 g_k 和实极点 c_k。而二阶因式表示复共轭零点 h_k 和 h_k^*，以及共轭极点 d_k 和 d_k^*。当(5.1.1)式中的所有系数都为实数时，式(5.1.4)表示了该系统的零点和极点的分布。该式代表由一阶和二阶子系统级联组成的一组结构形式。而我们知道，一个 N 阶系统函数可以用它的零、极点表示。由于 H(z)的系数均为实数，因此零、极点只有两种可能，或者为实数，或者为复共轭对。则整个系统函数可以完全分解成实系数二阶因子的形式，即

$$H(z)=A\prod_{k=1}^{N_c}\frac{1+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}} \tag{5.1.5}$$

其中 N_c 表示(N+1)/2 的最大整数。在这种情况下我们已经假设 M≤N，在将 H(z)写成这种形式时，假设实数极点和实数零点已经成对合并，并具有奇数个零点，则系统 b_{2k} 有一个等于零。同样，如果具有奇数个极点，则系统 a_{2k} 有一个等于零。从前面对直接形式结构的讨论可知，如果每个二阶子系统用直接形式 II 实现，就可以得到具有最少存储的级联结构。

一个四阶系统的级联结构如图 5.4 所示：

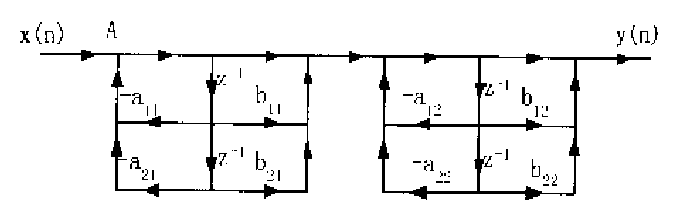


图5.4　四阶IIR数字滤波器的级联结构

在 MATLAB 中，给定直接形式滤波器的系数，可以计算出相应级联结构的各系数。

程序 1)　直接型滤波器转换为级联型滤波器。

```
function [b0,B,A]=dir2cas(b,a);
%变直接形式为级联形式
%[b0,B,A]=dir2cas(b,a)
%b0=增益系数
%B=包含各因子系数 bk 的 K 行 3 列矩阵
%A=包含各因子系数 ak 的 K 行 3 列矩阵
%a=直接型分子多项式系数
%b=直接型分母多项式系数

%计算增益系数
b0=b(1);b=b/b0;
```

```
a0=a(1);a=a/a0;
b0=b0/a0;
%将分子、分母多项式系数的长度补齐进行计算
M=length(b);N=length(a);
if N>M
   b=[b zeros(1,N-M)];
elseif M>N
   a=[a zeros(1,M-N)];N=M;
else
   NM=0;
end
%级联型系数矩阵初始化
K=floor(N/2);B=zeros(K,3);A=zeros(K,3);
if K*2==N
   b=[b 0];
   a=[a 0];
end
%根据多项式系数利用函数 roots 求出所有的根
%利用函数 cplxpair 进行按实部从小到大的成对排序
broots=cplxpair(roots(b));
aroots=cplxpair(roots(a));
%取出复共轭对的根变换成多项式系数即为所求
for i=1:2:2*K
   Brow=broots(i:1:i+1,:);
   Brow=real(poly(Brow));
   B(fix(i+1)/2,:)=Brow;
   Arow=aroots(i:1:i+1,:);
   Arow=real(poly(Arow));
   A(fix(i+1)/2,:)=Arow;
End
```

仍对 5.1.2 节的例子进行计算，在 MATLAB 中输入如下：

```
[b0,B,A]=dir2cas(b,a);
b0 =
   0.0625
B =
   1.0000   -0.0000    9.0000
   1.0000   -3.0000    2.0000
A =
   1.0000    1.0000    0.5000
   1.0000   -0.2500   -0.1250
```

所以滤波器的级联形式为：

$$H(z)=0.0625\frac{(1+9z^{-2})(1-3z^{-1}+2z^{-2})}{(1+z^{-1}+0.5z^{-2})(1-0.25z^{-1}-0.125z^{-2})}$$

程序 2)　级联型滤波器的实现。

```
function y=casfiltr(b0,B,A,x);
%滤波器的级联实现
%[y]=casfiltr(b0,B,A,x);
```

```
%b0=增益系数
%B=包含各因子系数 bk 的 K 行 3 列矩阵
%A=包含各因子系数 ak 的 K 行 3 列矩阵
%x=输入序列

[K,L]=size(B);
N=length(x);
w=zeros(K+1,N);
w(1,:)=x;
for i=1:K
%对于每个子系统而言，与直接型机构计算一样
   w(i+1,:)=filter(B(i,:),A(i,:),w(i,:));
end
y=b0*w(K+1,:);
```

仍对上例计算，在 MATLAB 中输入如下：

```
y=casfiltr(b0,B,A,x);
```

可见输出波形与用 filter 函数计算结果相同。

程序 3)　级联型滤波器转换为直接型滤波器。

```
function [b,a]=cas2dir(b0,B,A);
%变级联形式为直接形式
%[b,a]=dir2cas(b0,B,A)
%b0=增益系数
%B=包含各因子系数 bk 的 K 行 3 列矩阵
%A=包含各因子系数 ak 的 K 行 3 列矩阵
%a=直接型分子多项式系数
%b=直接型分母多项式系数
%初始化矩阵
[K,L]=size(B);
b=[1];
a=[1];
%将多项式项乘所得的系数即为所求
for i=1:K
   b=conv(b,B(i,:));
   a=conv(a,A(i,:));
end
b=b*b0;
```

对于用 dir2cas 求得的结果，可以进行如下计算：

```
[b1,a1]=casdir(b0,B,A);
y1=filter(b1,a1,x);
```

输出波形与图 5.3 一样，说明直接型和级联型的系统响应是相同的。

5.1.4 并联型

作为系统函数的另一种表示形式，可以将 H(z)表示如下形式的部分分式展开：

$$H(z)=\sum_{k=0}^{N_1}G_k z^{-k}+\sum_{k=0}^{N_2}\frac{A_k}{1-d_k z^{-1}}+\sum_{k=1}^{N_2}\frac{B_k(1-e_k z^{-1})}{(1-p_k z^{-1})(1-p_k^* z^{-1})} \tag{5.1.6}$$

由于(5.1.1)中的 H(z)的系数为实数，因此，式中各系数均为实数。如果 M≤N，则在(5.1.6)中不包括$\sum_{k=0}^{N}G_k z^{-k}$项。上式可以解释为一阶和二阶系统的并联组合。如果将实数极点成对组合，则可写成：

$$H(z)=\sum_{k=0}^{N_1}G_k z^{-k}+\sum_{k=1}^{N_p}\frac{e_{0k}+e_{1k}z^{-1}}{1-a_{1k}z^{-1}+a_{2k}z^{-2}} \tag{5.1.7}$$

对应的信号流图如图 5.5 所示(只取四阶)就是并联结构。

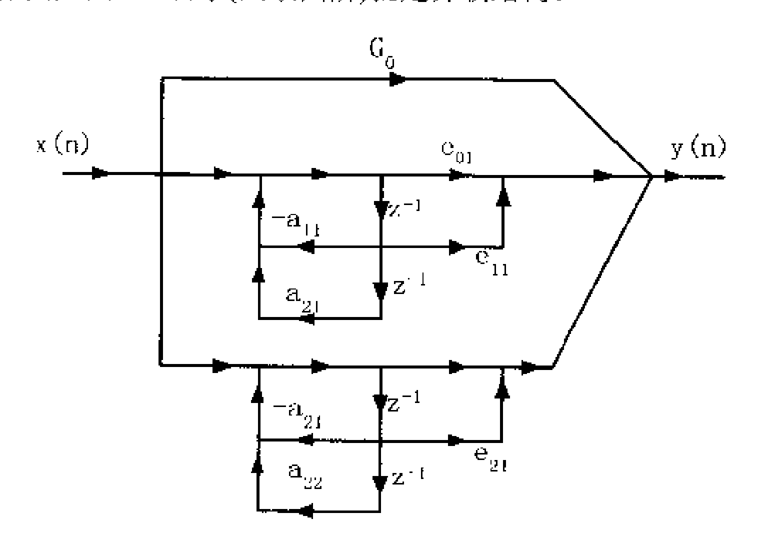


图5.5　四阶IIR数字滤波器的并联型结构

程序 4)　直接型滤波器转换为并联型滤波器。

```
function [C,B,A]=dir2par(b,a);
%变直接形式为并联形式
% [C,B,A]=dir2par(b,a)
%C=当分子多项式阶数大于分母多项式阶数时产生的多项式
%B=K 列 3 行 bk 系数矩阵
%A=K 列 3 行 ak 系数矩阵
%a=直接型分子多项式系数
%b=直接型分母多项式系数

M=length(b);N=length(a);
%将多项式分式分解为多项式与一阶多项式分式的和的形式
[r1,p1,C]=residuez(b,a);
%p1 即为系统的极点
p=cplxpair(p1,10000000*eps);
%重新排序后，就不一定能够将一阶多项式分式的分母分子正好对应，必须对分子系数排序
%函数 cplxcomp 见后
I=cplxcomp(p1,p);
r=r1(I);
```

```
%初始化系数矩阵
K=floor(N/2);B=zeros(K,2);A=zeros(K,3);
if K*2==N
%N 是偶数，则 A(z)的阶数是奇数，一个因子是一阶的
%将因子合并成二阶多项式分式
  for i=1:2:N-2
     Brow=r(i:1:i+1,:);
     Arow=p(i:1:i+1,:);
     [Brow, Arow]=residuez(Brow,Arow,[]);
     B(fix((i+1)/2),:)=real(Brow);
     A(fix((i+1)/2),:)=real(Arow);
  end
[Brow, Arow]=residuez(r(N-1),p(N-1),[]);
B(K,:)=[real(Brow') 0];
A(K,:)=[real(Arow') 0];
else
  for i=1:2:N-1
     Brow=r(i:1:i+1,:);
     Arow=p(i:1:i+1,:);
     [Brow,Arow]=residuez(Brow,Arow,[ ]);
     B(fix((i+1)/2),:)=real(Brow);
     A(fix((i+1)/2),:)=real(Arow);
  end
end
```

程序 5)　复共轭对比较。

```
function I=cplxcomp(p1,p2)
I=[];
for j=1:length(p2)
   for i=1:length(p1)
      if(abs(p1(i)-p2(j))<0.0001)
         I=[I,i];
      end
   end
end
I=I';
```

仍对上例计算，输入如下：

```
[C,B,A]=dir2par(b,a);
C =
   -18
B =
  -10.0500   -3.9500
   28.1125  -13.3625
A =
    1.0000    1.0000    0.5000
    1.0000   -0.2500   -0.1250
```

程序 6） 并联滤波器的实现。

```
function y=parfiltr(C,B,A,x);
%滤波器并联实现
%[Y]=parfiltr(C,B,A,x);
%C=当分子多项式阶数大于分母多项式阶数时产生的多项式
%B=K 列 3 行 bk 系数矩阵
%A=K 列 3 行 ak 系数矩阵
%x=输入序列

%初始化矩阵
 [K,L]=size(B);
N=length(x);
w=zeros(K+1,N);
w(1,:)=filter(C,1,x);
for i=1:K
%对于每个子系统而言，与直接型机构计算一样
   w(i+1,:)=filter(B(i,:),A(i,:),x);
end
y=sum(w)
```

仍对上例而言，在 MATLAB 中输入如下：

```
y=parfiltr(C,B,A,x);
```

输出波形与图 5.3 一样，说明直接型和并联型的系统响应是相同的。

程序 7） 并联型滤波器转换为直接型滤波器。

```
function [b,a]=par2dir(C,B,A);
%变并联形式为直接形式
%[b,a]=par2dir(C,B,A);
%C=当分子多项式阶数大于分母多项式阶数时产生的多项式
%B=K 列 3 行 bk 系数矩阵
%A=K 列 3 行 ak 系数矩阵
%a=直接型分子多项式系数
%b=直接型分母多项式系数
[K,L]=size(A);R=[];P=[];
for i=1:K
   [r,p,K]=residuez(B(i,:),A(i,:));
   R=[R;r];P=[P;p];
end
[b,a]=residuez(R,P,C);
b=b(:)';a=a(:)';
```

仍对上例而言，在 MATLAB 中输入如下：

```
[b1,a1]=par2dir(C,B,A);
y=filter(b1,a1)
```

输出波形与图 5.3 相同。

5.2　FIR 系统的结构

前面讨论的是针对实现无限冲激响应系统的。实现这样的系统必然要涉及递归计算法。对于有限冲激响应因果系统来说，它的实现一般是非递归的算法。有限长单位取样响应 FIR 滤波器突出特点是单位取样响应 h(n)仅有有限个非零值。即 h(n)为一个 N 点序列 $0 \leqslant n \leqslant N-1$，其中系统函数为：

$$H(z)=\sum_{n=0}^{N-1} h(n) z^{-n} \tag{5.2.1}$$

H(z)在 Z=0 处有 N－1 阶极点，而没有除 Z 平面原点(Z=0)外的极点。FIR 滤波器的结构主要是非递归结构；没有输出反馈，但有些结构，例如频率采样结构含有反馈的递归部分。同样 FIR 滤波器也有很多基本结构。

5.2.1　直接型

对于因果 FIR 系统，其系统函数只有多个零点(极点在原点处)，这相当于 IIR 滤波器系统函数 H(z)中所有系数 a_k 都为零，因此差分方程为：

$$y(n)=\sum_{k=0}^{M} b_k x(n-k) \tag{5.2.2}$$

其信号流图如图 5.6 所示。

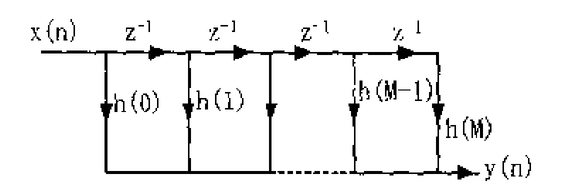


图5.6　FIR数字滤波器的直接结构

由图可见，这种结构与 IIR 数字滤波器在所有 a_k 系数等于零时的结构是一样的。故直接型 FIR 滤波器是 IIR 数字直接型滤波器的一种特殊情况。所以前面的有关直接型 IIR 滤波器的程序仍然适用于直接型 FIR 滤波器。

5.2.2　级联型

将 H(z)分解成实系数二阶因子的乘积形式：

$$H(z)=\sum_{n=0}^{N-1} h(n) z^{-n}=\prod_{k=1}^{N_c}\left(b_{0k}+b_{1k} z^{-1}+b_{2k} z^{-2}\right) \tag{5.2.3}$$

式中 N_c 是 N/2 的最大整数。这是因为 H(z)有奇数(N－1)个零点，其中复数零点成共轭对出现，且必为偶数个，故零点必有奇数个。其流图如图 5.7 所示。

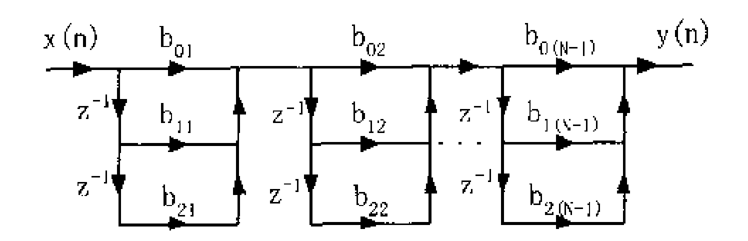


图5.7 FIR数字滤波器的级联结构

5.2.3 线性相位 FIR 系统结构

我们知道，若单位取样响应满足对称条件：

$$h(M-n)=h(n), n=0,1,\ldots,M \tag{5.2.4}$$

或

$$h(M-n)=-h(n), n=0,1,\ldots,M \tag{5.2.5}$$

则因果 FIR 系统就具有严格的线性相位。对于这种情况，系数乘法次数基本减少一半。为了证明这个条件满足线性相位的要求，可以将(5.2.1)式写成：

$$H(z)=\sum_{n=0}^{(N/2)-1}h(n)z^{-n}+\sum_{n=N/2}^{N-1}h(n)z^{-n}$$

$$\sum_{n=0}^{(N/2)-1}h(n)z^{-n}+\sum_{n=N/2}^{N-1}h(N-1-n)z^{-(N-1-n)} \tag{5.2.6}$$

式中假设 N 为偶数，利用(5.2.4)和(5.2.5)可以写出：

$$H(z)=\sum_{n=0}^{(N/2)-1}h(n)[z^{-n}+z^{-(N-1-n)}] \tag{5.2.7}$$

如 N 为奇数，容易证明：

$$H(z)=\sum_{n=0}^{[(N-1)/2]-1}h(n)[z^{-n}+z^{-(N-1-n)}]+h(\frac{N-1}{2})z^{-[(N-1)/2]} \tag{5.2.8}$$

若令 $z=e^{j\omega}$，计算(5.2.7)(5.2.8)，在 N 为偶数时得：

$$H(e^{j\omega})=e^{-j\omega[(N-1)/2]}\{\sum_{n=0}^{(N/2)-1}2h(n)\cos[\omega(n-\frac{N-1}{2})]\} \tag{5.2.9}$$

N 为奇数时得：

$$H(e^{j\omega})=e^{-j\omega[(N-1)/2]}\{\sum_{n=0}^{(N-3)/2}2h(n)\cos[\omega(n-\frac{N-1}{2})]+h(\frac{N-1}{2})\} \tag{5.2.10}$$

在这两种情况下，括号里的和式是实的，它蕴含着一个线性相移，这个相移相当于延迟 $\frac{N-1}{2}$ 个取样间隔。

M 是偶数时 FIR 线性相位系统的直接型结构流图如图 5.8 所示，M 是奇数时 FIR 线性相位系统的直接型结构流图如图 5.9 所示。

由于单位取样响应满足对称条件，使得 H(z)的零点以镜象对形式出现，如果 z_0 是 H(z)

的零点，则 $1/z_0$ 也是 H(z)的零点。又由于系数 h(n)是实数，故 H(z)的零点成复共轭对出现。所以，非单位圆上的实零点成反演对(倒数对)出现。非单位圆上的复零点成 4 个一组地出现，他们对应于复共轭和倒数关系。如果零点在单位圆上，其倒数又是其共轭，因此，单位圆上的复数零点通常成对地出现。

对于以上 3 种类型的 MATLAB 的实现，可以仍用前面介绍的几种扩展函数，只不过令 a=1。

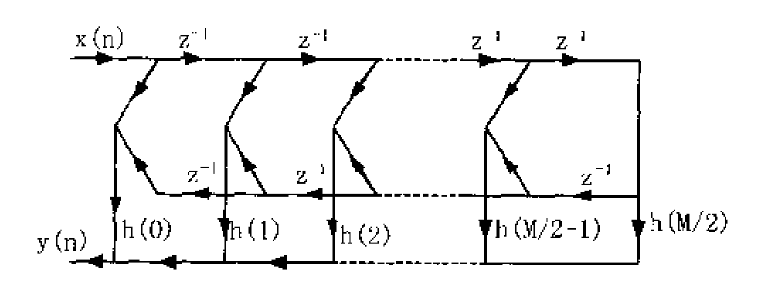


图5.8　M是偶整数时FIR线性相位系统的直接结构

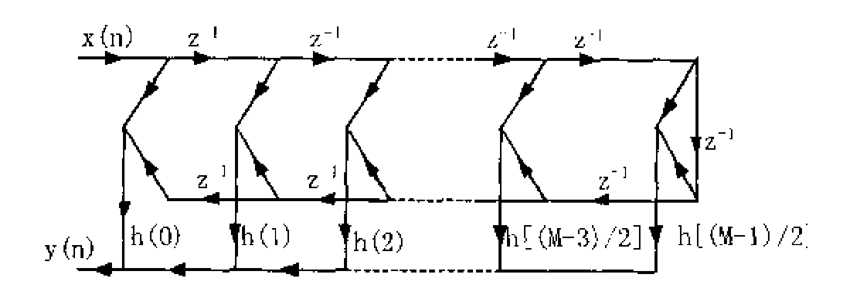


图5.9　M是奇整数时FIR线性相位系统的直接结构

5.2.4　频率取样型

一个有限时宽为 N 的序列其 z 变换可以用单位圆上的 N 个等间隔取样来表示。即系统函数 H(z)在单位圆上作 N 等分取样就是单位取样响应 h(n)的离散傅立叶变换 H(k)，H(k)和系统函数 H(z)之间的关系可用内插公式表示：

$$H(z)=(1/N)(1-z^{-N})\sum_{k=0}^{N-1}\frac{H(k)}{(1-W_N^{-k}z^{-1})} \tag{5.2.11}$$

将这个公式改写为：

$$H(z)=(1/N)H_F(z)\sum_{k=0}^{N-1}H_k(z) \tag{5.2.12}$$

由上式看出，FIR 系统可用一个子 FIR 系统和一个子 IIR 系统级联而成。其信号流图如图 5.10 所示。有限冲激响应系统的系统函数为 $1-z^{-n}$，它的零点在 z=exp[j(2π/N)k]。无限冲激响应部分由 N 个复一阶系统并联组成，极点在 $z_K=\exp[j(\frac{2\pi}{N})k]$ 上。这些一阶系统的极点准确的位于单位圆上，目的是正好对消有限冲激响应系统的一个零点。

一般来说，频率取样 H(k)和 W_k^{-N} 一样都是复数。因此，图 5.10 实现有限冲激响应系统需要复数运算。但当冲激响应取样 h(n)为实数，频率取样满足对称条件时：

$$|H(k)|=|H(N-k)| \tag{5.2.13}$$

$$\theta(k)=-\theta(N-k), k=0,1,\ldots,N-1 \tag{5.2.14}$$

则复一阶网络可以按复数共轭对合并而实现实系数的二阶网络。具体来说，假设 N 为偶数，频率取样可写成：

$$H(z)=\frac{1-z^{-N}}{N}[\sum_{k=1}^{(N/2)-1}\frac{H(k)}{1-W_N^{-k}z^{-1}}+\sum_{k=N/2+1}^{N-1}\frac{H(k)}{1-W_N^{-k}z^{-1}}+\frac{H(0)}{1-z^{-1}}+\frac{H(N/2)}{1+z^{-1}}] \tag{5.2.15}$$

改变第 2 个和式中的求和标号，上式变成

$$H(z)=\frac{1-z^{-N}}{N}[\sum_{k=1}^{(N/2)-1}(\frac{H(k)}{1-W_N^{-k}z^{-1}}+\frac{H(N-k)}{1-W_N^{-N+k}z^{-1}})+\frac{H(0)}{1-z^{-1}}+\frac{H(N/2)}{1+z^{-1}}] \tag{5.2.16}$$

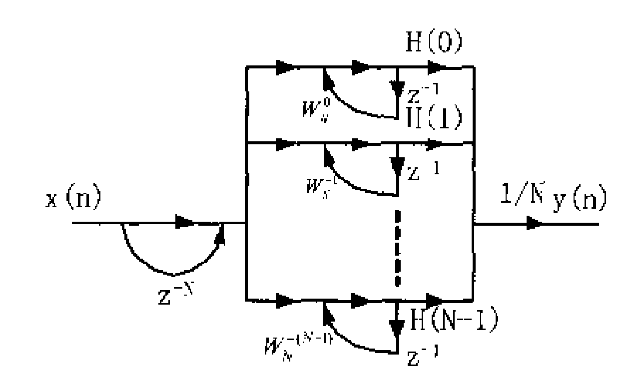


图5.10 FIR滤波器的频率取样型结构

利用对称条件，并考虑到$(W_N^{-k})^*=W_N^{-(N-k)}$，可将上式写为：

$$H(z)=(1-z^{-N})[\sum_{k=1}^{(N/2)-1}\frac{2|H(k)|}{N}H_k(z)+\frac{H(0)/N}{1-z^{-1}}+\frac{H(N/2)/N}{1+z^{-1}}] \tag{5.2.17}$$

式中：

$$H_k(z)=\frac{\cos(\theta(k))-z^{-1}\cos[\theta(k)-2\pi k/N]}{1-2z^{-1}\cos(2\pi k/N)+z^{-2}} \tag{5.2.18}$$

当 N 为奇数时，k=N/2 没有频率取样，因此上式不包含 H(N/2)的项。频率取样结构的主要优点是每一个二阶系统的输出端的乘数都正比于单位圆上等角度间隔上的频率响应取样。当然，这些值可以从冲激响应的离散傅立叶变换得到。如果实现的滤波器是具有一个或多个阻带的选频滤波器，它可以设计成阻带内的频率取样等于零，从而减少了必须实现的二阶系统 H_K 的数目。如果大部分的频率取样等于零(如窄带低通或带通滤波器)，则频率取样所需的乘法次数比直接形式少。其次，滤波器的结构的极点和零点仅取决于冲激响应的长度。如果输入用有限冲激响应滤波器组(即长度为 N 的几个不同的冲激响应)处理，则所有滤波器的一阶因式和二阶环节都可以用同一结构实现。

用 MATLAB 实现如下：

程序 8)　直接型滤波器转换为频率取样型滤波器。

```
function [C,B,A]=dir2fs(h);
%变h(n)值形式为频率取样形式
%[C,B,A]=dir2fs(h)
%C=并联部分的增益列向量
%B=按列排列的分子系数
%A=按列排列的分母系数
%h=FIR 滤波器的冲激响应

%计算 FIR 滤波器冲激响应的频率采样
M=length(h);
H=fft(h,M);
magH=abs(H);
phaH=angle(H)';
%判断M的奇偶性
if(M==2*floor(M/2))
   L=M/2-1;
   A1=[1,-1,0;1,1,0];
   C1=[real(H(1)),real(H(L+2))];
else
   L=(M-1)/2;
   A1=[1,-1,0];
   C1=[real(H(1))];
end
k=[1:L]'
B=zeros(L,2);A=ones(L,3);
A(1:L,2)=-2*cos(2*pi*k/M);A=[A;A1];
B(1:L,1)=cos(phaH(2:L+1));
B(1:L,2)=-cos(phaH(2:L+1)-(2*pi*k/M));
C=[2*magH(2:L+1),C1]';
```

例，令 h(n)={1,2,3,2,1}/9，确定并画出 FIR 的频率取样形式。

首先由 MATLAB 的 dir2fs 函数，将 h(n)直接形式转换成频率取样形式。输入如下：

```
h=[1,2,3,2,1]/9;
[C,B,A]=dir2fs(h);
C =
    0.5818
    0.0849
    1.0000
B =
   -0.8090    0.8090
    0.3090   -0.3090
A =
    1.0000   -0.6180    1.0000
    1.0000    1.6180    1.0000
    1.0000   -1.0000    0
```

然后，由于 M=5 为奇数，因此有：

$$H(z)=\frac{1-z^{-5}}{5}[0.5818\times\frac{-0.809+0.809z^{-1}}{1-0.618z^{-1}+z^{-2}}+0.0849\times\frac{0.309-0.309z^{-1}}{1+1.618z^{-1}+z^{-2}}+\frac{1}{1-z^{-1}}]$$

则可得出 FIR 的频率取样形式的信号流图，如图 5.11 所示。

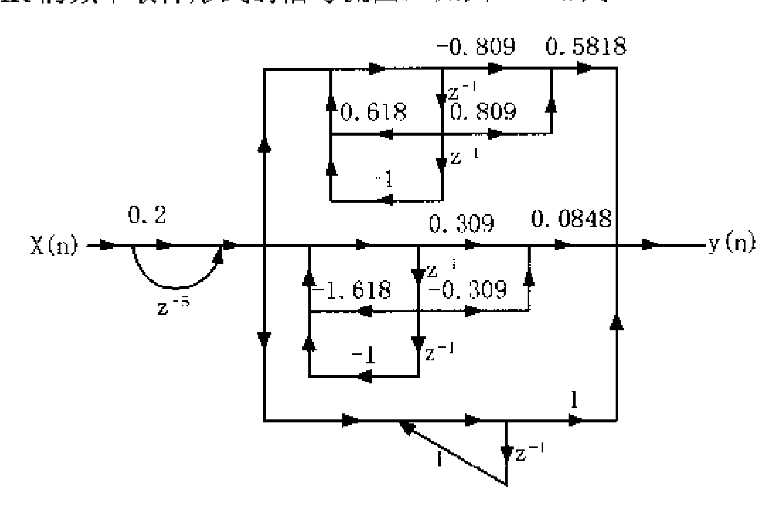


图5.11 FIR的频率取样形式(M=5)

5.3 离散系统的 Lattice 结构

1973 年，Gay 和 Markel 提出了一种新的系统的结构形式，即 Lattice 结构(又称格型结构)。事实证明，这是一种很有用的结构，在功率谱估计、语音处理、自适应滤波等方面已得到了广泛的应用。先分别讨论零点系统和全极点系统。

5.3.1 全零点系统 FIR 的 Lattice 结构

一个 M 阶的 FIR 系统的转移函数 H(z)可写为：

$$H(z)=B(z)=\sum_{i=0}^{M}b(i)z^{-i}=1+\sum_{i=1}^{M}b_M^{(i)}z^{-1} \tag{5.3.1}$$

系数 $b_M^{(i)}$ 表示 M 阶 FIR 系统的第 i 个系数，式中假定 H(z)=B(z)的首项系数等于 1，该系统的 Lattice 结构如图 5.12 所示。

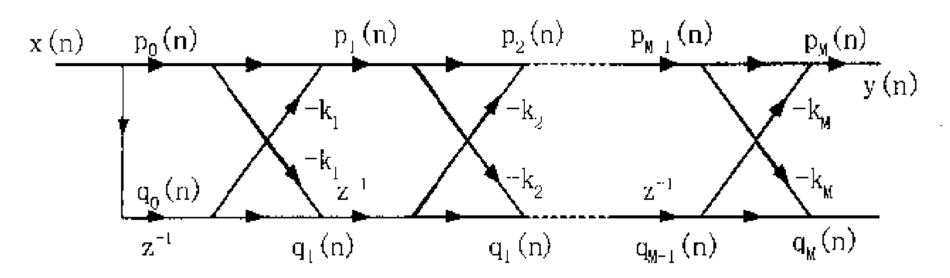


图5.12 FIR系统的Lattice结构

由该信号流图可以总结出 Lattice 结构的一些重要特点：

- H(z)的直接实现形式有 M 个参数，即 b(1)，b(2)，…，b(M)，共需 M 次乘法，M 次延迟；
- 信号的传递从左至右，中间没有反馈回路，所以这是一个 FIR 系统。若输入是 $\delta(n)$，则 $\delta(n)$通过信号流图的上部将立即出现在输出端，使 y(0)=h(0)=1。$\delta(n)$通过下部时，分别经过一次延迟、二次延迟，直到 M 次延迟后出现在输出端，所以 y(n)(即 h(n))的值是 y(1),…Y(M)；
- 信号流图中的基本单元有如下关系：

$$\begin{aligned} p_m(n) &= p_{m-1}(n) - k_m q_{m-1}(n-1) \\ q_m(n) &= -k_m p_{m-1}(n) + q_{m-1}(n-1), m = 1,2\ldots,M \end{aligned} \tag{5.3.2}$$

并且

$$\begin{aligned} p_0(n) &= q_0(n) = x(n) \\ y(n) &= p_M(n) \end{aligned} \tag{5.3.3}$$

- 若定义：

$$\begin{aligned} B_m(z) &= P_m(z)/P_0(z) = 1 + \sum_{i=1}^{m} b_m^{(i)} z^{-i}, m = 1,2,\ldots,M \\ \widetilde{B}_m(z) &= Q_m(z)/Q_0(z), m = 1,2,\ldots,M \end{aligned} \tag{5.3.4}$$

  那么 $B_m(z)$、$\widetilde{B}_m(z)$ 分别是由输入端 x(n)至第 m 个基本单元后所对应系统的转移函数，$B_m(z)$ 对应上端输出，$\widetilde{B}_m(z)$ 对应下端输出。当 m=M 时，$B_m(z)$=B(z)。显然，$B_m(z)$ 是 $B_{m-1}(z)$ 再级联上一个基本单元后所构成的较高一级的 FIR 系统。因此，Lattice 结构有着非常规则的结构形式。

对(5.3.2)式两边作 z 变换，有：

$$\begin{aligned} P_m(z) &= P_{m-1}(z) - k_m z^{-1} Q_{m-1}(z) \\ Q_m(z) &= -k_m P_{m-1}(z) + z^{-1} Q_{m-1}(z) \end{aligned} \tag{5.3.5}$$

将上述两式分别除以 $P_0(z)$、$Q_0(z)$，再由(5.3.4)式的定义，有：

$$\begin{bmatrix} B_m(z) \\ \widetilde{B}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m z^{-1} \\ -k_m & z^{-1} \end{bmatrix} \begin{bmatrix} B_{m-1}(z) \\ \widetilde{B}_{m-1}(z) \end{bmatrix} \tag{5.3.6}$$

即

$$\begin{bmatrix} B_m(z) \\ \widetilde{B}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & k_m \\ zk_m & z \end{bmatrix} \begin{bmatrix} B_{m-1}(z) \\ \widetilde{B}_{m-1}(z) \end{bmatrix} /(1-k_m^2) \tag{5.3.7}$$

上面两式给出了在格型结构中的由低阶到高阶或由高阶到低阶转移函数的递推关系，但这种递推中同时包含有 $B_m(z)$ 和 $\widetilde{B}_m(z)$。不难推出：

$$\widetilde{B}_m(z) = z^{-m} B_m(z^{-1}) \tag{5.3.8}$$

将该式分别代入(5.3.6)和(5.3.7)，有：

$$\begin{aligned} B_m(z) &= B_{m-1}(z) - k_m z^{-m} B_{m-1}(z^{-1}) \\ B_{m-1}(z) &= [B_m(z) + k_m z^{-m} B_m(z)]/(1-k_m^2) \end{aligned} \tag{5.3.9}$$

这样就分别得到了由高阶到低阶，或从低阶到高阶转移函数的递推关系。这种递推关系

中仅含有 $B_m(z)$。

利用上面得到的关系式可以得出如下递推关系：

$$\begin{aligned} &b_m^{(m)} = -k_m \\ &b_m^{(i)} = b_{m-1}^{(i)} - k_m b_{m-1}^{(m-i)} \\ &k_m = -b_m^{(m)} \\ &b_{m-1}^{(i)} = [b_m^{(i)} + k_m b_m^{(m-i)}]/(1-k_m^2) \end{aligned} \qquad (5.3.10)$$

在上式中，i=1,2,3,…,(m-1),m=1,2,3,…,M。

在实际工作中，一般是首先给出 H(z)=B(z)=B_m(z)，那么，可按如下步骤求出格型滤波器的系数：

(1) 由上述关系，首先得到 $k_M = -b_M^{(M)}$；

(2) 有(5.3.9)式，由 k_M 即系数 $b_M^{(1)}$，$b_M^{(2)}$，…$b_M^{(M)}$，求出 B_{M-1}(z)的系数；

(3) 重复步骤 2，$k_M, k_{M-1}, \ldots, k_1, B_{M-1}(z), \ldots, B_1(z)$ 可以全部求出。

可以用 MATLAB 语言实现将 FIR 的直接形式转换为全零点格型结构。

程序 9) 直接型滤波器转换为格型滤波器。

```
function [K]=dir2latc(b)
%[K]=dir2latc(b);
%K=格型滤波器的系数矩阵
%b=FIR 滤波器的直接形式系数矩阵
M=length(b);
K=zeros(1,M);
b1=b(1);
if b1==0
   error('b(1) is equal to zero')
end
K(1)=b1;A=b/b1;
for m=M:-1:2
   K(m)=A(m);
   j=fliplr(A);
   A=(A-K(m)*j)/(1-K(m)*K(m));
   A=A(1:m-1);
End
```

可以用 MATLAB 语言实现将全零点格型结构转换为 FIR 的直接形式。

程序 10) 格型滤波器转换为直接型滤波器。

```
function [b]=latc2dir(K);
%b=FIR 直接形式的系数
%K=格型结构的系数
M=length(K);
J=1;A=1;
for m=2:M
   A=[A,0]+conv([0,K(m)],J);
   J=fliplr(A);
end
b=A*K(1);
```

FIR 滤波器的格型结构的实现

程序 11)　格型滤波器的实现。

```
function [y]=latcfilt(K,x);
%y=输出序列
%K=格型结构滤波器的系数矩阵
%x=输入序列
Nx=length(x)-1;
M=length(K)-1;K=K(2:M+1);
fg=[x;[0 x(1:Nx)]];
for m=1:M
   fg=[1,K(m);K(m),1]*fg;
   fg(2,:)=[0 fg(2,1:Nx)];
end
y=fg(1,:);
```

例，一个 FIR 系统的零点分别在 $0.9e^{\pm j\frac{\pi}{3}}$ 及 0.8 处，求其格型结构。

解　$H(z)=B(z)=(1-z^{-1}0.9e^{j\frac{\pi}{3}})(1-z^{-1}0.9e^{-j\frac{\pi}{3}})(1-0.8z^{-1})$

$$=1-1.7z^{-1}+1.53z^{-2}-0.648z^{-3}$$

在 MATLAB 中输入：

```
b=[1,-1.7,1.53,-0.648]
k=dir2latc(b)
```

可得到：

```
k =[ 1.0000   -0.7026    0.7385   -0.6480]
```

于是该系统的格型结构如图 5.13 所示。

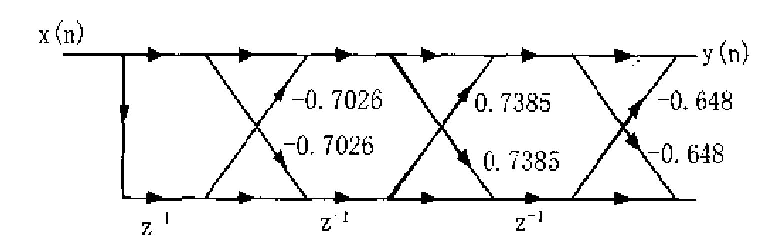


图5.13　系统的格型结构

例，FIR 滤波器的差分方程为：

$y(n)=2x(n)+\frac{13}{12}x(n-1)+\frac{5}{4}x(n-2)+\frac{2}{3}x(n-3)$，确定其格型结构，并画出直接形式和格型形式的冲激响应。

解　在 MATLAB 中输入如下：

```
b=[2,13/12,5/4,2/3];
K=dir2latc(b);
[x,n]=impseq(0.0.30);
y1=filter(b,1,x);
y2=latcfilt(K,x);
```

```
figure(1)
subplot(2,1,1)
plot(n,y1)
title('Direct form')
subplot(2,1,2)
plot(n,y2)
title('Lattice form')
```

执行后得：

```
K=
   2.0000      0.2500      0.5000     0.3333
```

因此格型滤波器的系数为：

$k_0=2$, $k_1=1/4$, $k_2=1/2$, $k_3=1/3$

滤波器如图 5.14 所示。

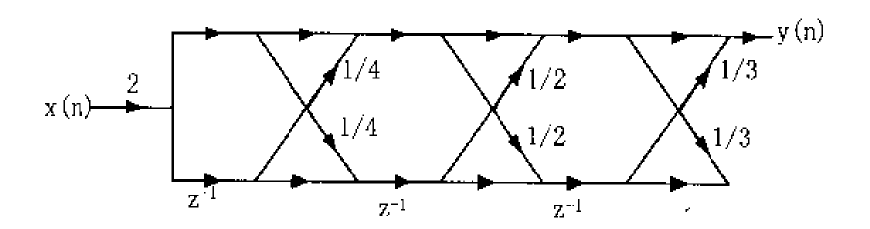


图5.14 FIR滤波器格型结构

则得到的冲激响应如图 5.15 所示，可以看出，两种形式的冲激响应是一样的。

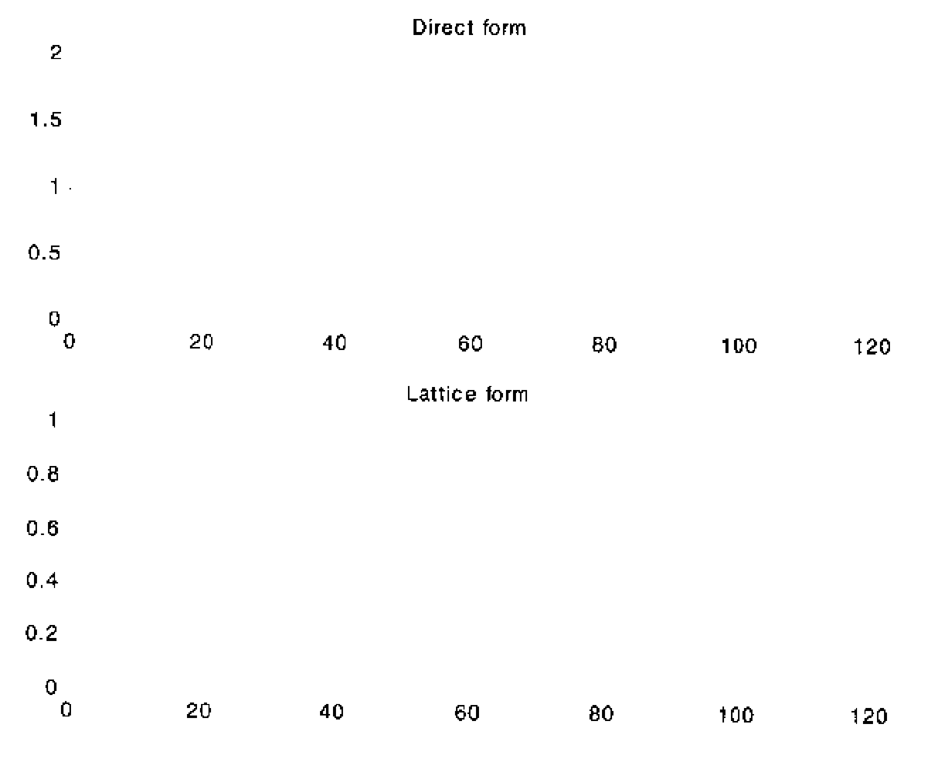


图5.15 FIR滤波器的冲激响应

5.3.2　全极点 IIR 系统的 Lattice 结构

对图 5.12 的 FIR 系统的格型结构基本单元，可重写(5.3.2)式：

$$\begin{aligned} p_{m-1}(n) &= p_m(n) + k_m q_{m-1}(n-1) \\ q_m(n) &= -k_m p_{m-1}(n) + q_{m-1}(n-1) \end{aligned} \tag{5.3.11}$$

这时，$p_m(n)$ 是上支路的输入信号，$p_{m-1}(n)$ 是输出信号，对下支路，$q_{m-1}(n-1)$ 是输入信号，$q_m(n)$ 是输出信号。设所给系统仍是 M 阶的多项式，并令 $x(n)=p_M(n)$，$p_0(n)=q_0(n)=y(n)$，则可得如图 5.16 所示的格型结构。

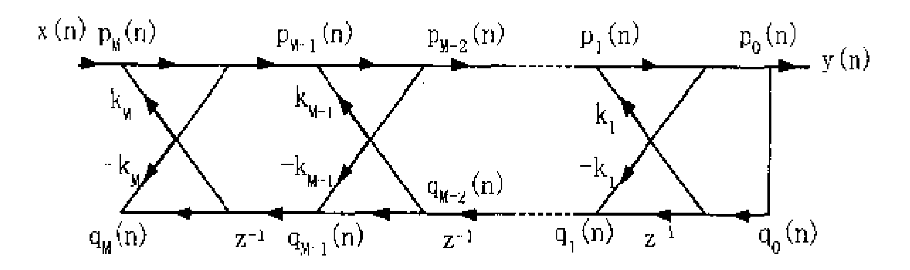


图5.16　全极点系统的格型结构

现在可以利用前面的推导方法来导出对应的转换函数及参数的求解方法。对应的一阶格型结构有：

$$\begin{aligned} p_0(n) &= p_1(n) + k_1 q_0(n-1) \\ q_1(n) &= -k_1 p_0(n) + q_0(n-1) \end{aligned} \tag{5.3.12}$$

由于 $p_0(n)=q_0(n)=y(n)$，$p_1(n)=x(n)$，所以上式又可写成：

$$\begin{aligned} y(n) &= p_1(n) + k_1 y(n-1) = x(n) + k_1 y(n-1) \\ q_1(n) &= -k_1 y(n) + y(n-1) \end{aligned} \tag{5.3.13}$$

这样，(5.3.13)式表示 x(n)为输入、y(n)为输出的一阶 IIR 系统和 $q_1(n)$为输出、$y(n)(q_0(n))$为输入时的一阶 FIR 系统，若令：

$$\frac{Y(z)}{P_1(z)} = \frac{1}{1-k_1 z^{-1}} = \frac{1}{A_1(z)} \tag{5.3.14}$$

则

$$\frac{Q_1(z)}{Y(z)} = -k_1 + z^{-1} = z^{-1}(1-k_1 z) = z^{-1}A_1(z^{-1}) = \widetilde{A}_1(z) \tag{5.3.15}$$

若令 M=2，则

$$\begin{aligned} p_1(n) &= p_2(n) + k_2 q_1(n-1) \\ q_2(n) &= -k_2 p_1(n) + q_1(n-1) \end{aligned} \tag{5.3.16}$$

这时，$p_2(n)=x(n)$，再由(5.3.12)式有：

$$\begin{aligned} y(n) &= k_1(1-k_2)y(n-1) + k_2 y(n-2) + x(n) \\ q_2(n) &= -k_2 y(n) - k_2(1-k_2)y(n-1) + y(n-2) \end{aligned} \tag{5.3.17}$$

则表示一个二阶 IIR 系统和一个二阶 FIR 系统，再令：

$$\frac{Y(z)}{P_2(z)}=\frac{1}{A_2(z)} \qquad \frac{Q_2(z)}{Y(z)}=\tilde{A}_2(z) \tag{5.3.18}$$

则

$$A_2(z)=1-k_1(1-k_2)z^{-2}-k_2z^{-2}=z^{-2}A_2(z^{-1})=\tilde{A}_2(z) \tag{5.3.19}$$

由此类推，若定义：

$$\frac{Y(z)}{P_m(z)}=\frac{1}{A_m(z)} \qquad \frac{Q_m(z)}{Y(z)}=\tilde{A}_m(z) \tag{5.3.20}$$

则

$$\tilde{A}_m(z)=z^{-m}A_m(z^{-1}) \tag{5.3.21}$$

且

$$H(z)=\frac{Y(z)}{X(z)}=\frac{Y(z)}{P_M(z)}=\frac{1}{A_M(z)}=\frac{1}{1+\sum_{i=1}^{M}a_M^{(i)}z^{-i}} \tag{5.3.22}$$

这样，图 5.12 对应的是一个全极点的 IIR 系统的格型结构，它正好是图 5.16 的逆过程。由于两个结构的最基本的差分方程是一样的，所以系数的求解方法同 FIR 系统格型结构的计算方法是一样的，区别只是将多项式的系数 b_m^i 换成 a_m^i 。

例，考虑全极点 IIR 滤波器 $H(z)=\dfrac{1}{1+\frac{13}{24}z^{-1}+\frac{5}{8}z^{-2}+\frac{1}{3}z^{-3}}$，确定其格型滤波器结构。

解　在 MATLAB 中输入如下：

```
a=[1,13/24,5/8,1/3]
K=dir2latc(a)
```

执行后得：

```
K=
   1.0000,    0.2500,    0.5000,    0.3333
```

因此：

$k_1=1/4,\quad k_2=1/2,\quad k_3=1/3$

k_0默认为 1。其格型结构如图 5.17 所示。

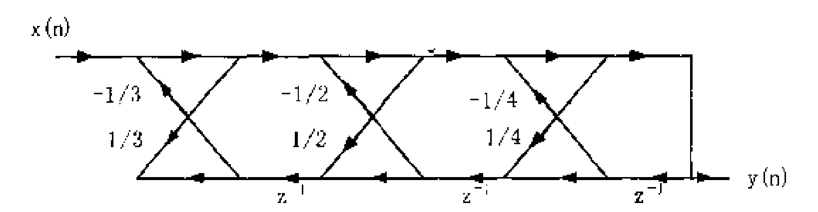


图5.17　IIR滤波器格型结构

第6章 基于MATLAB的 IIR DF设计

6.1 数字滤波器的基本原理

数字滤波器(Digital Filter，简称为DF)是数字信号处理的重要基础，在对信号的过滤、检测与参数的估计等信号处理中，数字滤波器是使用最为广泛的一种线性系统。

数字滤波器是对数字信号实现滤波的线性时不变系统。数字滤波实质上是一种运算过程，实现对信号的运算处理。输入的数字信号(数字序列)通过特定的运算转变为输出的数字序列，因此，数字滤波器本质上是一个完成特定运算的数字计算过程，也可以理解为是一台计算机。描述离散系统输出与输入关系的卷积和差分方程只是给数字滤波器提供运算规则，使其按照这个规则完成对输入数据的处理。

在前面的章节中我们讨论了时域离散系统的频域特性：

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$$

其中 $Y(e^{j\omega})$、$X(e^{j\omega})$ 分别是数字滤波器的输出序列和输入序列的频域特性(或称为频谱特性)，$H(e^{j\omega})$ 是数字滤波器的单位取样响应的频谱，又称为数字滤波器的频域响应。可以看出，输入序列的频谱 $X(e^{j\omega})$ 经过滤波后变为 $X(e^{j\omega})H(e^{j\omega})$，因此，只要按照输入信号频谱的特点和处理信号的目的，适当选择 $H(e^{j\omega})$，使得滤波后的 $X(e^{j\omega})H(e^{j\omega})$ 满足设计的要求，这就是数字滤波器的滤波原理。

线性数字滤波器和模拟滤波器(Analog Filter，简写为AF)一样，按照频域响应的通带特性可划分为低通、高通、带通、带阻几种形式。他们的理想模式如图6所示(只表示正频部分)。按照奈奎斯特采样定理，频率特性只能限带于 $|\omega| < \pi$ 的范围。

数字滤波器按照单位取样响应 $h(n)$的时域特性可分为无限脉冲响应(Infinite Impulse Response, IIR)系统和有限脉冲响应(Finite Impulse Response, IIR)系统。如果单位取样响应是时宽无限的：$h(n)$ $n_v \leqslant n < \infty$，这称之为IIR系统。而如果单位取样响应是时宽有限的：$h(n)$ $n_v \leqslant n < \infty$，这称之为FIR系统。

数字滤波器按照实现的方法和结构形式分为递归型或非递归型两类。递归型数字滤波器的当前输出y(n)值为输入x(n)的当前值和以前各输入值x(n-1)，x(n-2)，…，及以前的各个输出值y(n-1)，y(n-2)，…的函数。

一个N阶递归型数字滤波器的差分方程为：

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{i=1}^{N} a_i y(n-i) \tag{6.1.1}$$

由递归术语的含义，(6)式中的系数 a_i 至少有一项不为零。$a_i \neq 0$ 说明必须将延时的输出序列 y(n-i)反馈回来。因此从结构上看递归系统必须有反馈环路。

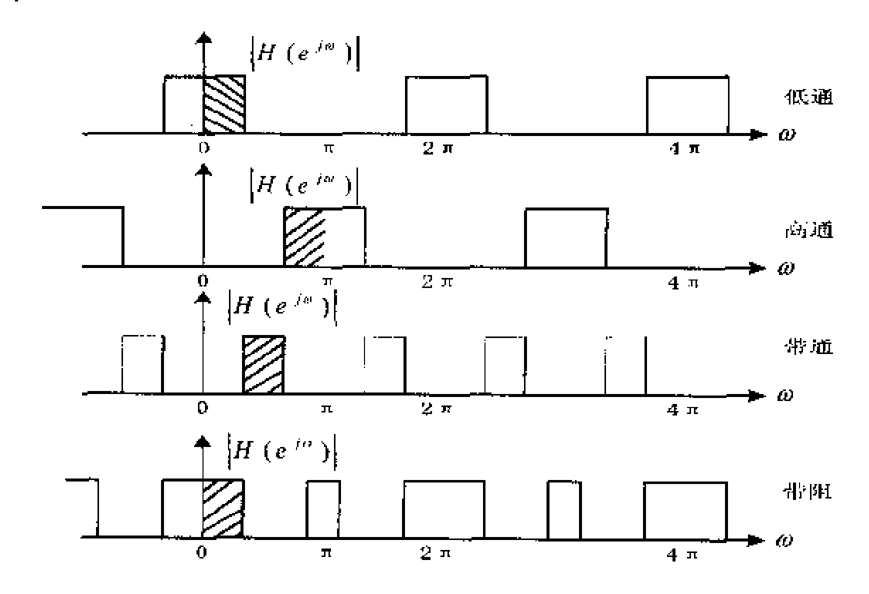


图6.1 数字滤波器的理想幅频特性模式

相应的系统函数为：

$$H(z)=\frac{\sum_{i=0}^{M} b_i z^{-i}}{1+\sum_{i=1}^{N} a_i z^{-i}} \tag{6.1.2}$$

递归系统的系统函数 H(z)在 z 平面上不仅有零点，而且有极点。

非递归型数字滤波器当前的输出值 y(n)仅为当前的和以前的输入序列的函数，而与以前的各个输出值无关，因此，从结构上看非递归系统没有反馈环路。一个 N 阶非递归型数字滤波器的差分方程是卷积形式：

$$y(n)=\sum_{k=0}^{N-1} h(k)x(n-k)=\sum_{k=0}^{N-1} b_k x(n-k) \tag{6.1.3}$$

差分方程的系数 b_k 等于单位取样响应序列值 h(n)。其系统函数 H(z)可以表示为以下形式：

$$H(z)=\sum_{K=0}^{N-1} b_k z^{-k} \tag{6.1.4}$$

H(z)是 z^{-1} 的多项式，因此它的极点只能在 z 平面的原点上。

通常，IIR 数字滤波器利用递归结构比较容易实现，而 FIR 数字滤波器利用非递归结构比较容易实现，但在 FIR 系统中也可以含有递归支路。一般将 IIR 数字滤波器与递归型、FIR 数字滤波器与非递归型联系在一起。

设计数字滤波器包括以下几个步骤：

(1) 按照实际任务的要求，确定滤波器的性能指标。

(2) 用一个因果、稳定的离散线性时不变系统的系统函数去逼近这一性能指标。根据

不同的要求可以用 IIR 系统函数，也可以用 FIR 系统函数去逼近。

(3) 利用有限精度算法实现系统函数。这里包括结构的选择、字长选择等。

在本章和第 7 章中，我们将讨论的重点放在第 2 部分，并且阐述如何利用 MATLAB 设计、实现、分析数字滤波器。

6.2　常用模拟滤波器的设计

为了由模拟滤波器设计 IIR 数字滤波器，首先设计一个满足技术指标的模拟原型滤波器。即把数字滤波器的技术指标转换成模拟滤波器的技术指标进行设计，设计得到模拟的原型滤波器。在这一节中将介绍模拟低通滤波器的设计，其中包括巴特沃斯、切比雪夫和椭圆滤波器的设计。

设计一个滤波器，重要的是寻找一个稳定、因果的系统函数去逼近滤波器的技术指标。一个因果、稳定的模拟滤波器的系统函数 $H_a(s)$应该满足如下条件：

- 滤波器的单位冲击响应函数 $h_a(t)$应该是一个实函数，即 $H_a(s)$是一个具有实系数的 s 的有理函数。
- $H_a(s)$的极点必须分布在 s 平面的左半平面。
- $H_a(s)$的分子多项式的阶数必须小于或等于分母多项式的阶数。

模拟滤波器的幅度响应通常采用“幅度平方函数” $|H_a(j\Omega)|^2$ 表示，根据上述的 3 个特点，$|H_a(j\Omega)|^2$ 必有以下特征，$|H_a(j\Omega)|^2$ 可以表示为：

$$|H_a(j\Omega)|^2 = H_a(j\Omega)H^*{}_a(j\Omega) \tag{6.2.1}$$

由于 $h_a(t)$ 为实函数，因此 $H_a(j\Omega)$ 满足：

$$\begin{aligned} &H^*{}_a(j\Omega) = H_a(-j\Omega) \\ &|H_a(j\Omega)|^2 = H_a(j\Omega)H_a(-j\Omega) = H_a(s)H_a(s)\big|_{s=j\Omega} \end{aligned} \tag{6.2.2}$$

式中的 $H_a(s)$ 是模拟滤波器的系统函数，它是 s 的有理函数。$H_a(j\Omega)$ 为滤波器的频率响应特性。

我们怎样从幅度平方函数确定系统函数呢？

从 6.2.2 式中得知，$H_a(s)$有一个极点(或零点)位于 $s = s_0$ 处，则 $H_a(-s)$在 $s = -s_0$ 处必有一个极点(或零点)。因为 $h_a(t)$为实函数，则 $H_a(s)$的极点(或零点)必然共轭出现。若 $H_a(s)$互为共轭的一对极点(或零点)位于$-a\pm jb$ 处，则 $H_a(-s)$必有一对$-a\mp jb$ 的极点(或零点)与之对应。显然，$H_a(s)$在 s = a 处的极点，$H_a(-s)$必在 s= −a 处有极点。当 a=0 的零点、极点位于虚轴上时，也必定是二阶或偶阶。因此 $H_a(s)H_a(-s)$的极点和零点分布如图 6.2 所示，零点和极点是对称分布的。

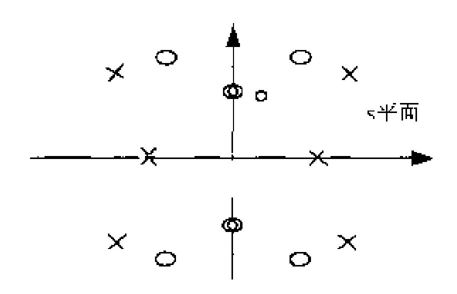


图6.2 Ha(-S)Ha(S)的零点的极点分布图

根据上面的条件2，$H_a(s)$的极点必须位于s左半平面，而右半平面的极点必须属于$H_a(-s)$。至于零点，则取决于设计的滤波器是否要求是最小相位的。如果要求是最小相位的，则$H_a(s)$的所有零点必须分布在左半平面或虚轴上。

综上所述，由$|H_a(j\Omega)|^2$来确定$H_a(s)$的方法是：

(1) 用$s=j\Omega$代入$|H_a(j\Omega)|^2$中得到$H_a(s)H_a(-s)$的函数。

(2) 将$H_a(s)H_a(-s)$因式分解，得到其全部的零点和极点，把左半平面的极点归于$H_a(s)$。若无特殊的要求，取$H_a(s)H_a(-s)$的对称零点的一半归于$H_a(s)$的零点。若要求设计最小相位的滤波器，则取左半平面的零点作为$H_a(s)$的零点。虚轴上的零点是偶次的，其中的一半归于$H_a(s)$。

(3) 由$H_a(s)$的低频特性，即$H_a(s)|_{s=0}=H_a(j\Omega|_{\Omega=0}$确定增益常数。

实际上设计模拟原型滤波器是要寻求一个逼近理想低通滤波器的函数，所谓原型低通滤波器是指低通模拟或数字滤波器。模拟低通滤波器的设计方法有：巴特沃斯、切比雪夫和椭圆滤波器的设计方法，下面将分别进行讨论。

6.2.1 巴特沃斯低通滤波器的设计

模拟低通巴特沃斯滤波器是一巴特沃斯函数作为滤波器的系统函数，它的幅度平方函数表示为：

$$|H_a(j\Omega)|^2=\frac{1}{1+\varepsilon^2\left(\frac{j\Omega}{j\Omega c}\right)^{2N}} \tag{6.2.3}$$

式中的N为正整数，表示滤波器的阶数。Ω_c为通带的截止频率，或$H_a(s)$的3分贝带宽。这是因为当$\Omega=\Omega_c$时：

$$|H_a(j\Omega)|^2=\frac{1}{2} \qquad (归一化后的巴特沃斯滤波器，\varepsilon=1)$$

$$|H_a(j\Omega)|=\frac{1}{\sqrt{2}}$$

$$20\log_{10}\left(\frac{|H_a(j0)|}{|H_a(j\Omega_c)|}\right)=3dB$$

半功率点的宽度等于3分贝带宽。巴特沃斯滤波器在通带有最平坦的幅度特性，即N

阶低通滤波器在 $\Omega=0$ 处的幅度平方函数 $|H_a(j\Omega)|^2$ 的前(2N-1)阶导数为零。阻带内随着频率的升高单调下降。滤波器的特征完全由阶数 N 决定。当 N 增加时，通带更加平坦，也更接近理想的低通滤波器的特性。

模拟的低通巴特沃斯滤波器的设计过程包括以下两个过程。

1. 按照给定的通带和阻带指标确定阶数N

通常在设计模拟低通滤波器时给定的技术指标有通带的截止频率 Ω_c，通带内最大衰减 $A_p(dB)$，阻带截止频率 Ω_s，阻带内的最小衰减 $A_s(dB)$。

假设给定 $\Omega=\Omega_c$ 时通带的最大衰减为 A_p，$\Omega=\Omega_s$ 时阻带的最小衰减为 A_s。低通巴特沃斯滤波器的幅度平方函数可以用图 6.3 表示。这时，通带的容限为：

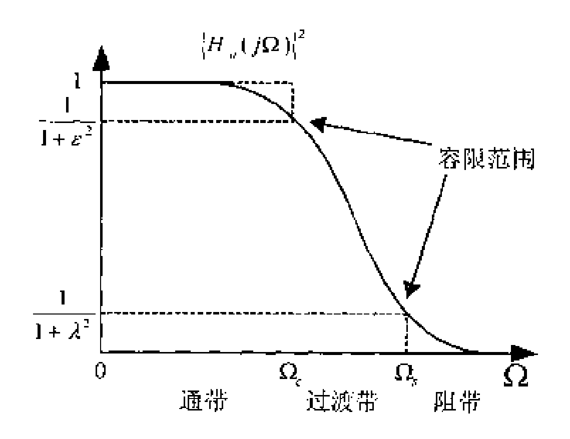


图6.3　低通巴特沃斯滤波器幅度平方函数

$$|H_a(j\Omega)|>\frac{1}{1+\varepsilon^2}\qquad |\Omega|\leqslant\Omega_c$$

阻带的的容限为：

$$|H_a(j\Omega)|<\frac{1}{1+\varepsilon^2}\qquad |\Omega|\geqslant\Omega_c$$

其中 ε、λ 均为与衰减有关的参数。

按照 6.2.3 式，可以得到：

$$\begin{aligned}A_p&=-10\log_{10}\left(\frac{1}{1+\varepsilon^2}\right)=10\log_{10}\left(1+\varepsilon^2\right)\\A_s&=-10\log_{10}\left(\frac{1}{1+\lambda^2}\right)=10\log_{10}\left(1+\lambda^2\right)\end{aligned}\tag{6.2.4}$$

由上二式可以求得：

$$\begin{aligned}\varepsilon&=\left(10^{0.1A_p}-1\right)^{0.5}\\\lambda&=\left(10^{0.1A_s}-1\right)^{0.5}\end{aligned}\tag{6.2.5}$$

由于

$$\varepsilon^2\left(\frac{j\Omega_s}{j\Omega_c}\right)^{2N}=\lambda^2$$

可以得到 N 为正整数的取值要求：

$$N \geqslant \frac{\log_{10}\left(\frac{\lambda}{\varepsilon}\right)}{\log_{10}\left(\frac{j\Omega_s}{j\Omega_p}\right)}$$

当给定的参数指标是归一化的，即 $\Omega_c=1$，$\varepsilon=1$ 时，可以表示为：

$$N \geqslant \frac{\log_{10}(\lambda)}{\log_{10}(\Omega_s)} \tag{6.2.6}$$

2. 从幅度平方函数确定系统函数 $H_a(s)$

令 s=jΩ 代入 6.2.1 式中，得到：

$$H_a(s)H_a(-s)=\frac{1}{1+\varepsilon^2\left(\frac{s}{j\Omega c}\right)^{2N}}$$

巴特沃斯滤波器的全部零点都在 s=∞ 处，即没有有限零点。故又称之为“全极点型”逼近。剩下的工作就是求解上式的分母多项式的根。

当考虑到设计归一化的巴特沃斯滤波器时，即 Ω$_c$=1，ε=1 时，它的极点应该均匀分布在单位圆上。得到 $H_a(s)$ 的极点为：

$$s_{pk}=-\sin\left(\frac{2k-1}{2N}\pi\right)+j\cos\left(\frac{2k-1}{2N}\pi\right)$$

$$k=\begin{cases}1,2,\cdots,\quad (N+1)/2 & N\text{为奇数}\\ 1,2,\cdots N/2 & N\text{为偶数}\end{cases} \tag{6.2.7}$$

如果归一化的系统函数用 $H_{aN}(s)$ 表示，同样，在左半平面上的极点组成 $H_{aN}(s)$，$H_{aN}(s)$ 和 $H_a(s)$ 的关系为：

$$H_a(s)=H_{aN}(s/\Omega_c)$$

例，设计一个巴特沃斯低通滤波器，满足以下技术指标：

通带的截止频率 $\Omega_c=10^4 rad/s$，通带的最大衰减 $A_p=3dB$，阻带的截止频率 $\Omega_s=4\times10^4 rad/s$，阻带的最小衰减为 $A_s=35dB$。

由式 6.2.5 可以求得：

$$\varepsilon=1,\qquad \lambda=56.2$$

然后由 6.2.6 式可以得到：

$$N \geqslant \frac{\log_{10}\left(\frac{\lambda}{\varepsilon}\right)}{\log_{10}\left(\frac{j\Omega_s}{j\Omega_p}\right)}=2.9$$

取 N=3 可以满足设计的要求。然后根据 6.2.5 可以得到 $H_a(s)$ 的极点的位置：

$$s_1=\Omega_c\left(-0.5+j\sqrt{3}/2\right)$$
$$s_1=-\Omega_c$$
$$s_1=\Omega_c\left(-0.5-j\sqrt{3}/2\right)$$

$H_a(s)$ 可以表示为极点的形式：

$$H_a(s)=\frac{1}{(s-s_1)\ (s-s_2)\ (s-s_3)}$$

最后可以得到满足系统设计指标的函数 $H_a(s)$：

$$H_a(s)=\frac{1}{\left(1+2\times10^{-2}s+2\times10^{-2}s^2+s^3\right)}$$

3. 利用MATLAB 设计模拟的低通巴特沃斯滤波器

在设计模拟的低通巴特沃斯滤波器的过程中可以利用 MATLAB 的函数 Buttap 进行滤波器的设计。Buttap 的语法为：

```
[Z, P, K]=buttap(N)
```

其中的 N 表示巴特沃斯滤波器的阶数，而函数的返回值 Z、P、K 分别表示滤波器的零点、极点和增益。对于上面这个例题可以通过一个 MATLAB 程序完成。

程序清单：

```
passrad=10000;
stoprad=40000;
passgain=3;
stopgain=35;
t1=sqrt(10^(0.1*passgain)-1);
t2=sqrt(10^(0.1*stopgain)-1);
n=ceil(log10(t2/t1)/log10(stoprad/passrad));
[z,p,k]=buttap(n);
syms rad;
hs1=k/(i*rad/passrad-p(1))/(i*rad/passrad-p(2))/…
(i*rad/passrad-p(3));
hs2=10*log10((abs(hs1))^2);
ezplot(hs2,[-60000,60000]);
grid on;
```

得到滤波器的极点的位置为：

```
-0.5000 + 0.8660i
-0.5000 - 0.8660i
-1.0000
```

滤波器的增益系数：

```
1.0000
```

得到的滤波器的幅度平方函数如图 6.4 所示，并且和理想的低通滤波器比较。其中的幅度平方利用分贝值表示。

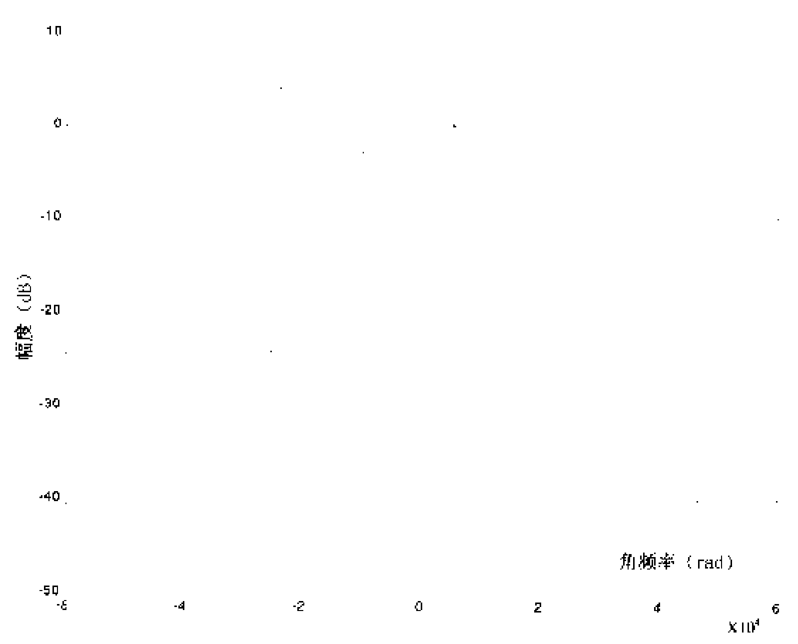


图6.4 巴特沃斯低通滤波器和理想低通滤波器幅度频率响应

6.2.2 切比雪夫低通滤波器的设计

巴特沃斯滤波器的频率特性在通带和阻带内部都是随着频率的单调变化的，显然，如果在通带的边缘能够满足指标，在通带的内部肯定能够超过设计的指标要求，造成滤波器的阶数 N 比较高。如果将指标的精度要求均匀地分布在整个通带内，或者均匀地分布在阻带内，更有效的方法是同时均匀地分布在通带和阻带内，可以设计出满足设计要求、阶数又比较低的滤波器。这就要求逼近函数具有等波纹特征。而切比雪夫 I 型滤波器在通带内幅度特性是等波纹的，在阻带是单调的。而切比雪夫 II 则相反，它在通带内是单调的，在阻带内是等波纹的。

I 型的切比雪夫低通滤波器的幅度平方函数为：

$$|H_a(j\Omega)|^2=\frac{1}{1+\varepsilon^2C_N^2\left(\dfrac{j\Omega}{j\Omega_c}\right)} \tag{6.2.8}$$

式 6.2.6 是一个正实函数。式中的 ε 是一个小于 1 的正数，它与通带波纹有关，ε 越大，波纹也越大。Ω_c 为通带的截止频率，$C_N(x)$ 时 N 阶切比雪夫多项式，定义为：

$$C_N(x)=\begin{cases}\cos(N\cos^{-1}(x)) & |x|\leqslant 1\\ ch(Nch^{-1}(x)) & |x|>1\end{cases} \tag{6.2.9}$$

当 N 大于或等于 1 时，切比雪夫多项式的递推公式为：

$$C_{N+1}(x)=NxC_N(x)-C_{N-1}(x)$$

切比雪夫滤波器由 3 个参数需要确定：ε、Ω_c 和 N，其中 Ω_c 是给定的截止频率。ε 由容许的通带波纹或通带的幅度误差 δ 确定。通常波纹误差 δ(dB)为：

$$\delta = 10\lg\frac{|H_a(j\Omega)|^2_{max}}{|H_a(j\Omega)|^2_{min}} = 20\lg\frac{|H_a(j\Omega)|_{max}}{|H_a(j\Omega)|_{min}}$$

由于通带幅度响应的最大值 $|H_a(j\Omega)|_{max}=1$，最小值 $|H_a(j\Omega)|_{min}=\frac{1}{\sqrt{1+\varepsilon^2}}$，由此得到：

$$\varepsilon^2 = 10^{0.1A_p}-1$$

$$\varepsilon = (10^{0.1A_p}-1)^{0.5}$$

其中 A_p 是通带内的最大衰减，单位是 dB。

滤波器的阶数 N 等于在通带内等幅波动的次数，即最大值和最小值的总数。当 N 为奇数时，由于 $|C_N(0)|=0$，则 Ω=0 处为最大值。当 N 为偶数时，$|C_N(0)|=1$，则 Ω=0 处为最小值。

ε 按照 6.2.6 式确定后，根据给定的阻带衰减确定 N 的数值。设阻带的截止频率为 Ω_s，当 $\Omega=\Omega_s$ 时，阻带的容许衰减为 A_s(dB)，则：

$$A_s = 10\lg\left(\frac{|H_a(0)|^2}{|H_a(j\Omega_s)|^2}\right)$$

因为 $H_a(0)=1$，所以：

$$A_s = -10\lg\left(|H_a(j\Omega_s)|^2\right) = 10\lg\left(1+\lambda^2\right)$$

$$\lambda = (10^{0.1A_s}-1)^{0.5}$$

阻带的幅度平方数值为：

$$|H_a(j\Omega)|^2 = \frac{1}{1+\varepsilon^2 C_N^2\left(\frac{j\Omega}{j\Omega_c}\right)} = \frac{1}{1+\lambda^2}$$

化简得到：

$$C_N\left(\frac{j\Omega}{j\Omega_c}\right) = \frac{\lambda}{\varepsilon}$$

又因为(Ω_s/Ω_c)>1，由 6.2.7 式得到：

$$C_N(j\Omega_s/j\Omega_c) = ch(Nch^{-1}(j\Omega_s/j\Omega_c)) = \frac{\lambda}{\varepsilon}$$

故，滤波器的阶数 N 为：

$$N \geq \frac{ch^{-1}(\frac{\lambda}{\varepsilon})}{ch^{-1}(\frac{\Omega_s}{\Omega_c})}$$

若要求阻带截止频率上的衰减越大，或过渡带内幅度特性越陡，则所需的阶数 N 越高。

切比雪夫滤波器的 $|H_a(j\Omega)|^2$ 也只有极点，而没有零点，且只需要计算出左半平面的极点。所以一旦 N，Ωc 与 ε 确定，$H_a(s)$ 也就确定了。

在 MATLAB 中，可以利用函数 cheb1ap 设计切比雪夫 I 型低通滤波器。Cheb1ap 的语法为：

```
[z,p,k]=cheb1ap(n,rp)
```

其中 n 为滤波器的阶数，rp 为通带的幅度误差。返回值分别为滤波器的零点、极点和增益。

例，设计满足下列技术指标的归一化的低通切比雪夫 I 型滤波器。

通带的最大衰减 A_p=1dB，阻带的截止频率 Ω_s=4rad/s，阻带的最小衰减为 40dB。

利用 MATLAB 进行设计，程序清单如下：

```
passrad=1;
stoprad=4;
passgain=1;
stopgain=35;
t1=sqrt(10^(0.1*passgain)-1);
t2=sqrt(10^(0.1*stopgain)-1);
n=ceil(acosh(t2/t1)/acosh(stoprad/passrad));
[z,p,k]=cheb1ap(n,passgain);
syms rad;
hs1=k/(i*rad -p(1))/(i*rad -p(2))/(i*rad -p(3));
hs2=10*log10((abs(hs1))^2);
ezplot(hs2,[-6,6]);
grid on;
```

得到滤波器的极点的位置为：

```
-0.2471 + 0.9660i
-0.4942 + 0.0000i
-0.2471 - 0.9660i
```

滤波器的增益系数：

```
0.4913
```

得到的滤波器的幅度平方函数如图 6.5 所示，并且和理想的低通滤波器比较。其中的幅度平方利用分贝值表示。

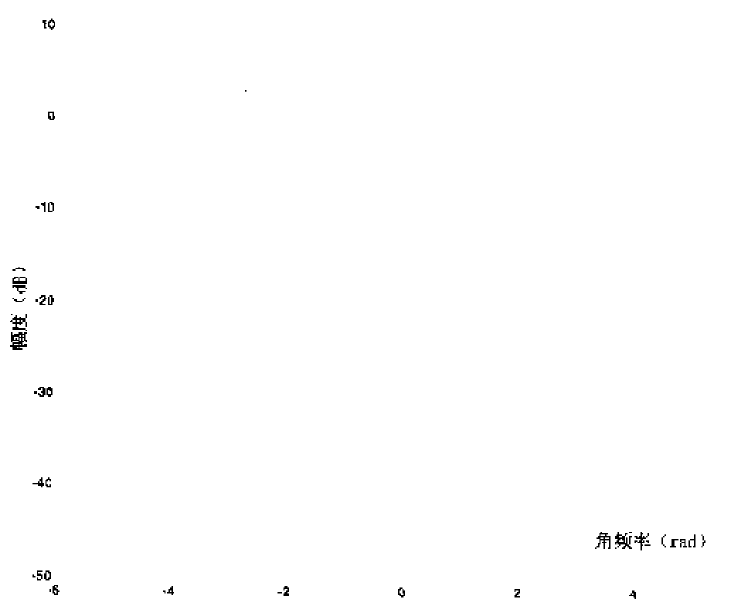


图6.5 切比雪夫低通滤波器和理想低通滤波器的幅度频率响应

6.2.3　椭圆低通滤波器的设计

切比雪夫滤波器在通带范围内具有良好的等波纹特性，但是在通带范围以外的特性与巴特沃斯滤波器相似，都是单调递减的。因此，过渡带的特性虽有好转，但是并不是理想的。它的主要原因在于二者的系统函数在截止频率的附近没有有限个零点，其零点在无限远处。考尔在 1931 年提出了采样有限零点设计的滤波器能够更好地逼近理想的低通滤波器特性。由于这种方法在确定零点的位置时与椭圆函数的许多特性有关，所以称之为椭圆低通滤波器。它的幅度平方函数为：

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\varepsilon^2 J_N^2(\Omega)} \tag{6.2.10}$$

式中的 $J_N(\Omega)$是雅可比椭圆函数，ε 是与通带衰减有关的参数，阶数 N 等于通带和阻带内最大点和最小点的总和。但知道通带的截止频率 Ω_c，通带内最大衰减 A_p(dB)和阻带的截止频率 Ω_s 及阻带内的最小衰减 A_s(dB)等参数时，可以确定阶数 N 和系统函数。

将 Ω_c 和 Ω_s 的几何平均值 Ω_0 作为频率的归一化的基准频率，即

$$\Omega_0 = \sqrt{\Omega_c \Omega_s}$$

定义频率的选择性因数 k 为两个截止频率之比：

$$k\frac{\Omega_c}{\Omega_s}$$

这时通带截止频率的归一化值和阻带截止频率的归一化值分别为：

$$\overline{\Omega}_c = \frac{\Omega_c}{\Omega_0} = \sqrt{k}$$

$$\overline{\Omega}_s = \frac{\Omega_s}{\Omega_0} = 1\Big/\sqrt{k}$$

在以 Ω_0 为归一化基准频率的条件下，通带、阻带阶值归一化频率互为倒数。现在令：

$$q_0 = \frac{1}{2}\frac{1-\left(1-k^2\right)^{1/4}}{1+\left(1-k^2\right)^{1/4}}$$

$$q = q_0 + 2q_0{}^5 + 15q_0{}^9 + 15q_0{}^{13}$$

$$b^2 = 10^{0.1A_s}$$

$$\varepsilon^2 = 10^{0.1Ap}$$

椭圆滤波器的阶数 N 为：

$$N \geqslant \frac{\log_{10}{}^{16}\left[{(b^2-1)}\big/{\varepsilon^2-1} \right]}{\log_{10}\left[1/q \right]}$$

一旦滤波器的阶数确定，令 Ω_0=1，则归一化的椭圆低通滤波器的系统函数可以表示为：

$$H_{aN}(s)=\frac{H_0}{D_0(s)}\prod_{i=1}^{M}\frac{s^2+A_i}{s^2+B_is+C_i}$$

其中 $M=\begin{cases}N/2 & N\text{为偶数}\\(N-1)/2 & N\text{为奇数}\end{cases}$

$$D_0(s)=\begin{cases}1 & N\text{为偶数}\\s+\sigma_0 & N\text{为奇数}\end{cases}$$

式中的各个参数都可以通过已知条件得到，因为表达式比较复杂，这里不加讨论。而实际的椭圆低通滤波器可以通过归一化的系统函数转化得到：

$$H_a(s)=H_{aN}(s/\Omega_0)$$

在 MATLAB 中，可以利用函数 ellipap 设计椭圆低通滤波器。ellipap 的语法为：

```
[z,p,k]=ellipap(n,rp,rs)
```

其中 n 为滤波器的阶数，rp 为通带的幅度误差，rs 为阻带内的最小衰减。返回值分别为滤波器的零点、极点和增益。

例，设计一个低通的椭圆滤波器，它的技术指标与前一小节切比雪夫滤波器设计的例题相同。

程序清单：

```
passrad=1;
stoprad=4;
passgain=1;
stopgain=35;
rad0=sqrt(passrad*stoprad);
k=passrad/stoprad;
normalpassrad=passrad/rad0;
normalstoprad=stoprad/rad0;
q0=(1-(1-k*k)^0.25)/(1+(1-k*k)^0.25)/2;
q=q0+2*q0^5+15*q0^9+15*q0^13;
b2=10^(0.1*stopgain);
e2=10^(0.1*passgain);
n=ceil(log10(16*((b2-1)/(e2-1)))/log10(1/q));
[z,p,k]=ellipap(n,passgain,stopgain);
syms rad;
hs1=k*(i*rad/rad0-z(1))*(i*rad/rad0-z(2))/...
    (i*rad/rad0-p(1))/(i*rad/rad0-p(2))/(i*rad/rad0-p(3));
hs2=10*log10((abs(hs1))^2);
ezplot(hs2,[-10,10]);
grid on;
```

得到滤波器的极点的位置为：

```
-0.2182 - 0.9810i
-0.2182 + 0.9810i
-0.5380
```

得到滤波器的零点的位置为：

```
0 - 2.3131i
```

```
0 + 2.3131i
```

滤波器的增益系数：

```
0.1016
```

得到的滤波器的幅度平方函数如图 6.6 所示，其中的幅度平方利用分贝值表示。

上面介绍了三种最常用的模拟滤波器的设计方法，选择其中的哪种形式的滤波器可以根据具体应用情况而定。通常椭圆滤波器的阶数最低，切比雪夫滤波器次之，巴特沃斯滤波器最高。而参数量化的灵敏度恰恰相反，巴特沃斯滤波器最佳(最不敏感)，切比雪夫次之，而椭圆滤波器对量化最敏感。

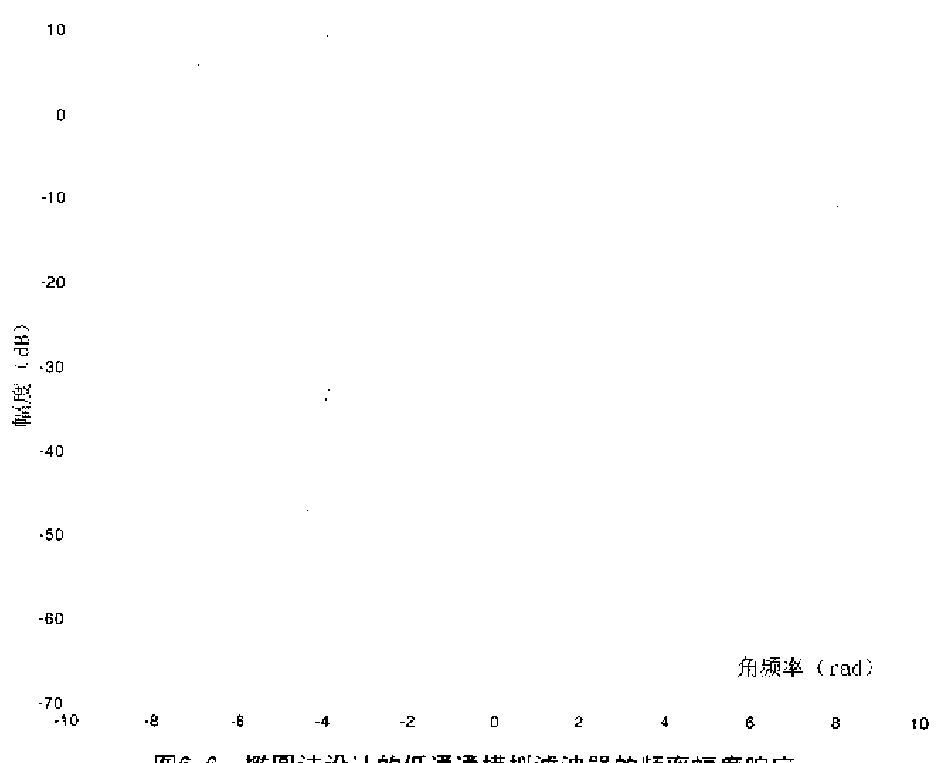


图6.6 椭圆法设计的低通通模拟滤波器的频率幅度响应

6.3 用脉冲响应不变法设计 IIR 滤波器

脉冲响应不变法的设计原理是使得数字滤波器的单位取样响应序列 h(n)模仿模拟滤波器的冲激响应 $h_a(t)$。

将模拟滤波器的冲激响应 $h_a(t)$进行等间隔采样，使得数字滤波器的单位取样响应 h(n)刚好等于 $h_a(t)$的采样值，即：

$$h(n)=h_a(t)\big|_{t=nT}=h_a(nT)$$

其中的 T 为采样周期。

若令 $H_a(s)$是模拟滤波器的系统函数，H(z)是数字滤波器的系统函数。显然，$H_a(s)$是 $h_a(t)$的拉普拉斯变换，而 H(z)是 h(n)的 z 变换。模拟信号的拉普拉斯变换与它的采样序列的 z 变换的关系为：

$$H(z)\Big|_{z=e^{sT}}=\frac{1}{T}\sum_{k=-\infty}^{\infty}H_a(s-j\frac{2\pi}{T}k)$$

可以看出，利用脉冲响应不变法将模拟滤波器变换成数字滤波器，实际上是首先将模拟滤波器的系统函数 $H_a(s)$作周期的延拓，在经过 $z=e^{sT}$ 的映射变换，从而得到数字滤波器的系统函数 H(z)。假设 s 平面上，s 在 jΩ 轴上取值，z 在 z 平面内的单位圆周 $e^{j\omega}$ 上取值，可以得到数字滤波器的频率响应 $H(e^{j\omega})$和模拟滤波器的频率响应 H(jΩ)间的关系为：

$$H(e^{j\omega})=\frac{1}{T}\sum_{k=-\infty}^{\infty}H_a(j\frac{\omega}{T}-j\frac{2\pi}{T}k)$$

但是对于任何一个实际的模拟滤波器，它的频率响应不可能是真正带限的。因而，将不可避免的出现频率的交叠，即混叠失真。数字滤波器的频率响应不能重现模拟滤波器的频率响应。只有当模拟滤波器的频率响应在超过折叠频率后的衰减很大时，混叠失真才很小，此时采样脉冲响应不变法设计的数字滤波器才能够满足设计的要求。

按照脉冲响应不变法的原理，用这种方法设计数字滤波器系统函数 H(z)的过程是：由模拟滤波器的系统函数 $H_a(s)$，求出它的拉普拉斯反变换得到脉冲响应 $h_a(t)$，然后对其进行等间隔采样：

$$h_a(t)\Big|_{t=nT}=h_a(nT)=h(n)$$

然后求出 h(n)的 z 变换，便得到系统函数 H(z)，即：

$$H_a(s)\to h_a(t)\to h(n)\to H(z)$$

通常按照上述的方法的过程比较繁琐，在实际中，脉冲响应不变法特别适合于模拟滤波器系统函数能够用部分分式展开式表示的情况。

假设模拟滤波器的系统函数 $H_a(s)$只有单阶极点，且 M<N，系统函数可以用部分分式形式表示：

$$H(s)=\sum_{k=1}^{N}\frac{A_k}{s-s_k}$$

其拉普拉斯变换为脉冲响应 $h_a(t)$为：

$$h_a(t)=\begin{cases}\sum_{k=1}^{N}A_k e^{s_k t} & t\geqslant 0\\ 0 & t<0\end{cases}$$

对 $h_a(t)$进行等间隔采样，可以得到数字滤波器的单位取样响应 h(n)：

$$h(n)=\begin{cases}\sum_{k=1}^{N}A_k e^{s_k nT} & t\geqslant 0\\ 0 & t<0\end{cases}$$

然后对 h(n)进行 z 变换，便得到数字滤波器的系统函数：

$$H(z)=\sum_{n=-\infty}^{\infty}h(n)z^{-n}$$
$$=\sum_{n=0}^{\infty}\sum_{k=1}^{N}A_k e^{s_k nT}z^{-n}$$
$$=\sum_{k=1}^{N}\frac{A_k}{1-e^{s_k T}z^{-1}}$$

由此可知，通过模拟滤波器的系统函数，可以直接求得数字滤波器的系统函数，这种方法求取数字滤波器的系统函数是比较方便的。

在 MATLAB 中，可以利用公式变换函数变化，得到数字滤波器的系统函数。

例，已知模拟低通滤波器的系统函数为：

$$H_a(s)=\frac{1}{s^2+3s+2}$$

利用冲激响应不变法设计数字低通滤波器，其中采样周期 T=1。

程序的清单：

```
syms s ht hn hs hz hrad2 hrad1 rad ;
hs=1/(s*s+3*s+2);
T=1;
ht=ilaplace(hs);
hn=subs(ht,n*T);
hz=ztrans(hn);
hrad1=subs(hs,i*rad);
hrad2=subs(hz,exp(i*rad/T));
ezplot(abs(hrad1),[-4*pi,4*pi]);
hold on;
ezplot(abs(hrad2),[-4*pi,4*pi]);
grid on;
```

通过计算得到数字滤波器的系统函数 h(z):

h(z)= z/exp(-1)/(z/exp(-1)-1)-z/exp(-2)/(z/exp(-2)-1)

数字滤波器的频率响应函数 $H(e^{j\omega})$：

exp(i*rad)/exp(-1)/(exp(i*rad)/exp(-1)-1)-exp(i*rad)/exp(-2)
/(exp(i*rad)/exp(-2)-1)

而模拟滤波器的频率响应为：

1/(-rad^2+3*i*rad+2)

图 6.7 显示了模拟滤波器和数字滤波器的频率幅度响应，数字滤波器的频率幅度响应是以 2π 为周期的函数，由于混叠失真的影响，数字滤波器的频率响应与模拟滤波器的幅度频率响应在 0～π 上有所不同。

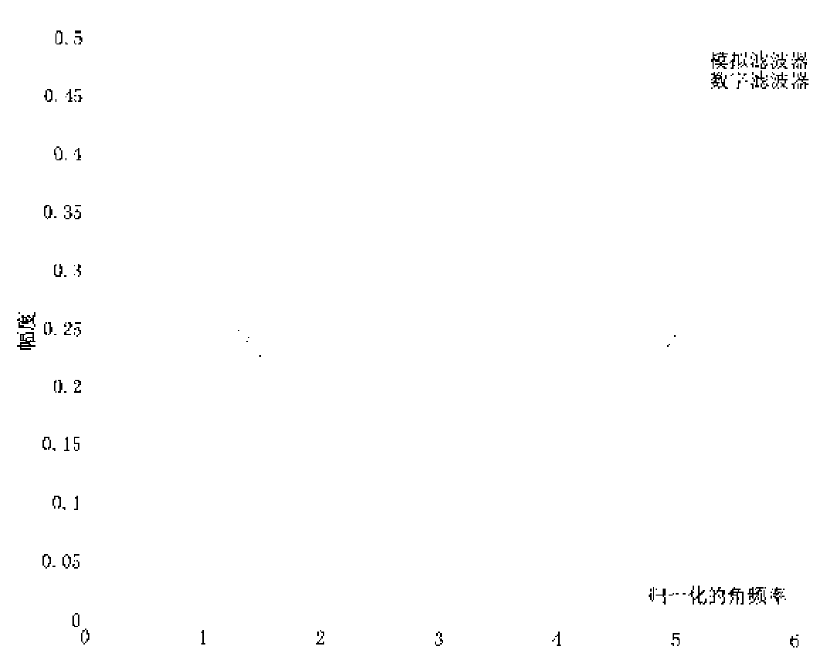


图6.7 利用冲击响应不变法设计的数字滤波器和原型滤波器的幅频特性

由于 $H_a(j\Omega)$在 $\Omega>\pi/T$ 处有明显的非零值，频谱交叠使得 $H(e^{j\omega})$ 有明显的失真。

脉冲响应不变化的主要特点是频率坐标的变换是线性变换，即：

$$\omega=\Omega T=\Omega/f_s$$

因而，如果模拟滤波器的频率响应是充分带限的，$H_a(j\Omega)=0$，当$|\Omega|>\frac{\pi}{T}$ 时。通过变换，数字滤波器的频率响应可以不失真地重现模拟滤波器的频率响应。

$$H(e^{j\omega})=H(e^{j\Omega T})=H_a(j\Omega)$$

$$|\Omega|<\frac{\pi}{T}$$

如果模拟滤波器是线性相位的低通滤波器，通过变换后，得到的数字滤波器仍然是线性相位的。

脉冲响应不变法的另一个特点是时域逼近良好，即脉冲响应不变法设计的数字滤波器的取样响应能够较好地模仿模拟滤波器的冲击响应，这在很多场合是非常需要的。

脉冲响应不变法最主要的缺点是由于频谱的周期延拓而产生的混叠失真。因而用这种方法设计的滤波器只适合于充分带限的低通或带通滤波器，而高通和带阻滤波器不宜采用脉冲响应不变法来设计。

6.4 用双线性变换法设计 IIR 滤波器

冲激响应不变法使得数字滤波器在时域上能够较好地模仿模拟滤波器，但是由于从 s 平面到 z 平面的映射 $z=e^{sT}$ 具有多值性，使得设计出来的数字滤波器不可避免的出现频谱的混叠。如果我们设想，若能够把整个 s 平面先映射到 s_1 平面的带域$\left(-\frac{\pi}{T}\leqslant\Omega_1\leqslant\frac{\pi}{T}\right)$，且使

得 s_1 平面的带域与 s 平面有单值对应关系，然后再利用 $z=e^{s_1T}$ 把 s_1 平面中的带域映射为 z 平面的整个平面上，且具有单值对应关系，消除多值性，使得 s 平面与 z 平面间建立一一对应的单值关系，从而消除了混叠现象。这便是双线性变换法的基本思路。

双线性变换化是使得数字滤波器的频率响应模仿模拟滤波器的频率响应的一种方法。这种方法的基本思路是：首先将整个 s 平面压缩到 s_1 平面的一条带宽为 2π/T(从－π/T 到 π/T)的横带里，然后通过标准的变换关系 $z=e^{s_1T}$将横带变换成整个 z 平面上去，这便得到 s 平面与 z 平面间的一一对应的单值关系。整个过程如图 6.8 所示。

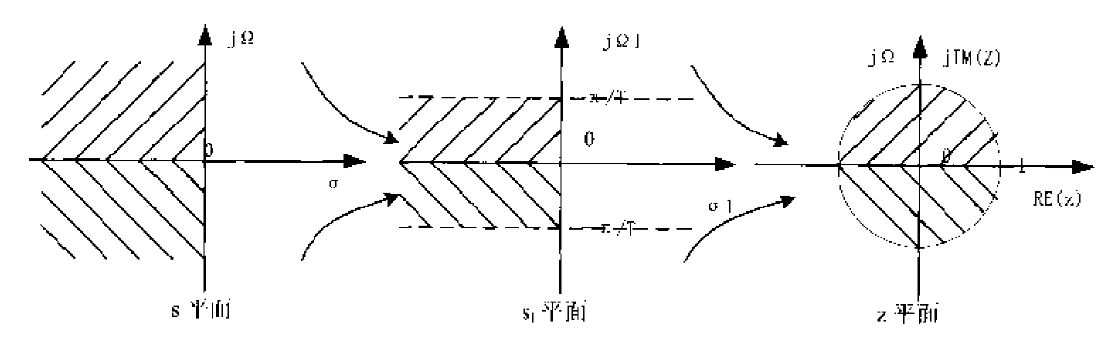


图6.8　双线性变换法的映射关系

为了将 s 平面的整个 jΩ 轴压缩变换到 s_1 平面 $j\Omega_1$ 轴上的－π/T 到 π/T 一段，采用了正切变换关系：

$$\Omega = k\tan\left(\Omega_1 T/2\right)$$

式中的 k 为常数。当 Ω_1 从－π/T 变到 π/T 时，从上式可知，Ω 便从－∞变到+∞，这样，s 平面内的整个 jΩ 轴便映射到 s_1 平面 $j\Omega_1$ 轴的一段：$-\pi/T\leqslant\Omega_1\leqslant\pi/T$。s 整个平面映射到 s_1 平面的一个带域内。

可以将上式改写成：

$$j\Omega = k\frac{e^{j\frac{\Omega_1}{2}T}-e^{-j\frac{\Omega_1}{2}T}}{e^{j\frac{\Omega_1}{2}T}+e^{-j\frac{\Omega_1}{2}T}}$$

解析延拓到整个 s 平面和 s_1 平面，即令 $j\Omega=s$，$j\Omega_1=s_1$，则：

$$s = k\frac{e^{\frac{s_1}{2}T}-e^{-\frac{s_1}{2}T}}{e^{\frac{s_1}{2}T}+e^{-\frac{s_1}{2}T}} = k\frac{1-e^{-s_1T}}{1+e^{-s_1T}}$$

由此得到了 s 平面和 s_1 平面的映射公式，采样脉冲响应不变法得到的标准映射公式为：

$$z=e^{s_1T}$$

将 s_1 平面映射到 z 平面。利用上面的两个公式可以得到从 s 平面到 z 平面的对应关系为：

$$s = k\frac{1-z^{-1}}{1+z^{-1}}$$

$$z = \frac{1+s/k}{1-s/k}$$

以上的两个公式是两个线性函数之比，故为线性分式变换，因此称之为算线性变换。其中的参数 k 可以根据模拟滤波器的频率特性和数字滤波器的频率特性在哪些频率点上具有的对应关系选择。例如，如果使得模拟滤波器与数字滤波器在低频处有比较确切的对应关系，即在低频处有 $\Omega \approx \Omega_1$。当 Ω 较小时，可以认为：

$$\tan\left(\Omega_1 T/2\right) \approx \Omega_1 T/2$$

从而可以得到：

$$\Omega_1 = k\,\Omega_1 T/2$$

解得 k=2/T。

通常在进行公式的运算过程中，过程比较繁琐，但是如果利用 MATLAB 的函数和符号处理工具可以很方便的解决。

例，已知模拟低通滤波器的系统函数为：

$$H_a(s) = \frac{1}{s^2 + 3s + 2}$$

利用冲激响应不变法设计数字低通滤波器，其中采样周期 T=1，数字滤波器的频率特性在低频处和模拟滤波器的频率特性相同。

程序清单：

```
syms  k hrad1 hrad0 rad0 rad1 hs hz sz s z;
T=1;
hs=1/(s*s+3*s+2);
k=2/T
sz=k*(1-1/z)/(1+1/z);
hz=subs(hs,sz);
hrad0=subs(hs,s,i*rad0);
hrad1=subs(hz,z,exp(i*rad1));
ezplot(abs(hrad1),[0,4*pi]);
hold on;
ezplot(abs(hrad0) ,[0,4*pi]);
grid on;
```

通过计算得到数字滤波器的系统函数 h(z)：

```
h(z)= 1/((2-2/z)^2/(1+1/z)^2+3*(2-2/z)/(1+1/z)+2)
```

数字滤波器的频率响应函数 $H(e^{j\omega})$：

```
1/((2-2/exp(i*rad1))^2/(1+1/exp(i*rad1))^2+3*(2-2/
exp(i*rad1))/(1+1/exp(i*rad1))+2)
```

而模拟滤波器的频率响应为：

```
1/(-rad^2+3*i*rad+2)
```

图 6.9 显示了模拟滤波器和数字滤波器的频率幅度响应，数字滤波器的频率幅度响应是以 2π 为周期的函数。

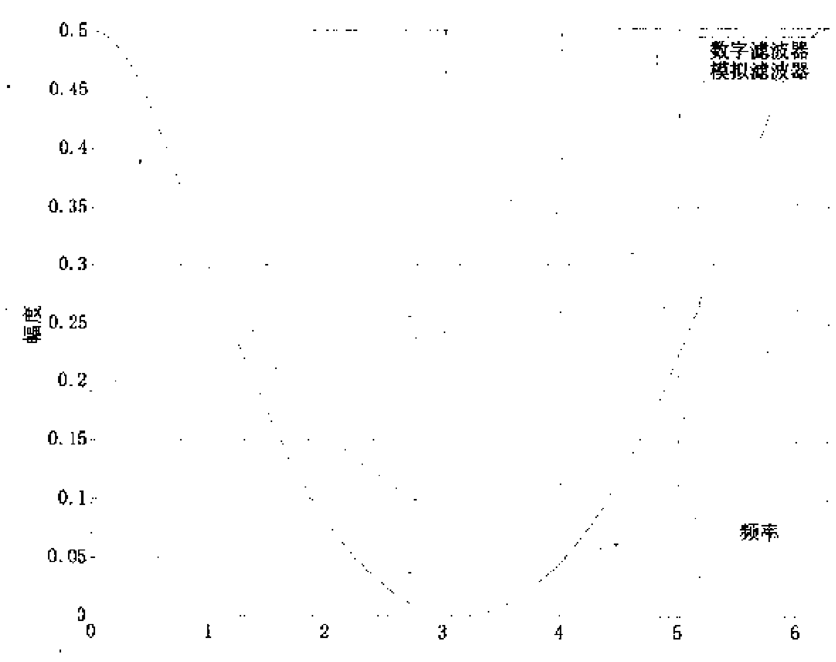


图6.9 利用双线性变换法得到的数字滤波器的幅频特性

双线性变换与脉冲响应不变法相比较，其主要优点是 s 平面与 z 平面之间是单一的一一对应关系。从而消除了频谱混叠现象。S 域变换到 z 域，双线性变换法不需要将模拟滤波器的系统函数进行分解，只需将系统函数 $H_a(s)$中的拉普拉斯算子 s 用 z 的函数代替，因而应用十分方便、简单。

但是这种方法有固有的 3 个缺点：模拟频率 Ω 和数字频率 ω 之间是非线性关系，它使得频率的标度弯曲；不能够保持原来的模拟滤波器的相频特性；H(z)的频率响应和时间响应与模仿的 $H_a(s)$有明显的差别。一般情况下，可以通过频率的预畸进行校正。

6.5 数字高通、带通及带阻滤波器设计

实际中通常要求的滤波器是低通、高通、带通、带阻、多带阻和多带通数字滤波器，前 4 种滤波器可以通过模拟的原型滤波器或数字的原型滤波器的频率转换得到，而后两种滤波器的转换方法还待进一步的研究；所有的数字滤波器可以通过直接法进行设计，也可以通过优化的方法得到。

在这一小节中，将讨论数字高通、带通、带阻以及低通滤波器的设计方法，概括来说，有 3 种设计方法，它们分别是：IIR 数字滤波器的原型转换设计法、IIR 数字滤波器的直接设计法、IIR 的优化设计法。在 6.6 节，我们将讨论在 MATLAB 中，利用 MATLAB 固有的函数直接设计各种形式的滤波器，其中包括巴特沃斯数字滤波器、切比雪夫 I 型数字滤波器、切比雪夫 II 型数字滤波器、贝塞尔方法设计的数字滤波器。

6.5.1 IIR 数字滤波器的原型转换设计法

所谓原型滤波器是指归一化的低通滤波器。如果数字滤波器由模拟低通滤波器转换得到，则可能有 3 种方法。一是首先由模拟的低通滤波器转换成数字低通滤波器，然后再用变量代换变换成所需的数字滤波器；二是由模拟的低通滤波器转换成所需的模拟滤波器，然后再把它转换成数字滤波器；三是由模拟低通滤波器直接转换成所需的数字滤波器。

下面将对上述 3 种转换方法分别进行讨论。

第 1 种方法：首先由模拟的低通滤波器首先转换成数字低通滤波器，然后再用变量代换变换成所需的数字滤波器。

这种方法的第 1 步在前面已经讨论过，它可以利用冲击响应不变法和双线性变换法实现。现在讨论数字原型滤波器到其他形式的数字滤波器的转换。转换应满足两点：一是转换后仍为稳定的因果系统，这就要求在原来 Z 平面上单位圆内部的映射到新的 Z 平面之后还在单位圆内部；二是要求转换后仍然是 z^{-1} 的有理函数。

转换关系若采用：

$$z^{-1}=G\left(Z^{-1}\right)$$

其中 z 表示原来的 z 平面，而 Z 表示新的 z 平面，则 $G(Z^{-1})$ 必须是 Z^{-1} 的有理函数，且 z 平面上单位圆内部必须映射到 Z 平面的单位圆的内部。用 $H_l(z)$ 表示原来滤波器的系统函数，$H_d(Z)$ 表示转换后的滤波器的系统函数，它们的关系可以表示为：

$$H_d\left(Z^{-1}\right)=H_1\left(G\left(Z^{-1}\right)\right)$$

式中：

$$Z^{-1}=G^{-1}\left(z^{-1}\right)$$

如果用 θ 、ω 分别表示 z 平面和 Z 平面的角频率，可以把 z、Z 表示成 $z=e^{j\theta}$、$Z=e^{j\omega}$ 的形式，则上式可以表示成：

$$e^{j\theta}=\left|G^{-1}\left(e^{-j\omega}\right)\right|e^{jatg\left(G^{-1}\left(e^{-j\omega}\right)\right)}$$

由等式两边相等得到 $\left|G^{-1}\left(e^{-j\omega}\right)\right|=1$，$\theta=-atg\left(G^{-1}\left(e^{-j\omega}\right)\right)$，因此 $G(Z^{-1})$ 的一般形式可以写成：

$$G\left(Z^{-1}\right)=\pm\prod_{k=1}^{N}\frac{Z^{-1}-a_k}{1-a_kZ^{-1}}$$

为了使得滤波器稳定，上式中的 a_k 的位置应该在单位圆内部，合适地选择 N 和 a_k 可以得到多种变换，其中最简单的变换是低通到低通的变换，此时：

$$z^{-1}=G\left(Z^{-1}\right)=\frac{Z^{-1}-a}{1-aZ^{-1}}$$

令 $z=e^{j\theta}$、$Z=e^{j\omega}$，则得到：

$$e^{j\theta}=\frac{e^{-j\omega}-a}{1-ae^{-j\omega}}$$

或

$$e^{-j\omega}=\frac{e^{-j\theta}+a}{1-ae^{-j\theta}}$$
$$=\frac{a+\cos\theta-j\sin\theta}{1+a\cos\theta-ja\sin\theta}$$
$$=\frac{2a+(1+a^2)\cos\theta-j(1-a^2)\sin\theta}{1+a^2+2a\cos\theta}$$
$$=1\times e^{-jarctg\left(\frac{(1-a^2)\sin\theta}{2a+(1+a^2)\cos\theta}\right)}$$

它的模等于 1，相角：

$$\omega=-arctg\left(\frac{(1-a^2)\sin\theta}{2a+(1+a^2)\cos\theta}\right)$$

由此可以得到 a 的表达式：

$$a=\frac{\sin\left(\left(\theta-\omega\right)/2\right)}{\sin\left(\left(\theta+\omega\right)/2\right)}$$

由此式可以看出，θ 和 ω 还是与 a 有关，可以用选择 a 值的办法把低通原型滤波器的 θ_p 的值映射到所需要滤波器的 ω_p 的值。当确定 θ_p 和 ω_p 之后，将这两个数值代入上式，可以计算出 a 的数值，由此可以得到新的数值滤波器的系统函数。

其他形式的滤波器的转换方法也完全类似，表 6.1 给出了变换公式和参数的计算公式。

表 6.1 滤波器的变换公式和参数计算公式

滤波器类型	$z^{-1}=G(Z^{-1})$	A 的计算公式
低通	$z^{-1}=\frac{Z^{-1}-a}{1-aZ^{-1}}$	$a=\frac{\sin\left(\left(\theta_p-\omega_p\right)/2\right)}{\sin\left(\left(\theta_p+\omega_p\right)/2\right)}$
高通	$z^{-1}=-\frac{Z^{-1}+a}{1+aZ^{-1}}$	$a=\frac{\cos\left(\left(\theta_p+\omega_p\right)/2\right)}{\cos\left(\left(\theta_p-\omega_p\right)/2\right)}$
带通	$z^{-1}=\frac{Z^{-2}-\frac{2ak}{k+1}Z^{-1}+\frac{k-1}{1+k}}{\frac{k-1}{1+k}Z^{-2}-\frac{2ak}{k+1}Z^{-1}+1}$	$a=\frac{\cos\left(\left(\omega_2+\omega_1\right)/2\right)}{\cos\left(\left(\omega_2-\omega_1\right)/2\right)}$ $k=\cot\left(\left(\omega_2-\omega_1\right)/2\right)tg\left(\frac{\theta_p}{2}\right)$
带阻	$z^{-1}=\frac{Z^{-2}-\frac{2ak}{k+1}Z^{-1}+\frac{1-k}{1+k}}{\frac{1-k}{1+k}Z^{-2}-\frac{2a}{k+1}Z^{-1}+1}$	$a=\frac{\cos\left(\left(\omega_2+\omega_1\right)/2\right)}{\cos\left(\left(\omega_2-\omega_1\right)/2\right)}$ $k=\cot\left(\left(\omega_2-\omega_1\right)/2\right)tg\left(\frac{\theta_p}{2}\right)$

例，依照首先由模拟的低通滤波器转换成数字低通滤波器，然后再用变量代换变换成所需的数字滤波器的设计思路，设计一数字高通滤波器。首先设计一个切比雪夫低通模拟滤波器，通带的截止频率为 1，通带的最大衰减 A_p=1dB，阻带的截止频率 1.5，阻带的最小衰减为 15dB。然后利用双线性变换法得到原型的低通滤波器，最后转换成高通的数字滤波器。高通滤波器的截止频率 ω_p=0.6 π。

程序清单：

```
syms rad hs1 hs2 hz1 hz2 ht hn sz z s rad1 rad2 n;
passrad=1;
stoprad=1.5;
passgain=1;
stopgain=15;
t1=sqrt(10^(0.1*passgain)-1);
t2=sqrt(10^(0.1*stopgain)-1);
m=ceil(acosh(t2/t1)/acosh(stoprad/passrad));
[z1,p,k]=cheb1ap(m,passgain);
hs1=k/(i*rad/passrad-p(1))/(i*rad/passrad-p(2))/...
(i*rad/passrad-p(3))/(i*rad/passrad-p(4));
T=1;
ht=ilaplace(hs1);
hn=subs(ht,n*T);
hz1=ztrans(hn);
angl1=0.6*pi;
angl2=passrad*T;
a=-cos((angl1+angl2)/2)/cos((angl1-angl2)/2);
z2=-(1+a/z)/(1/z+a);
hz2=subs(hz1,z2);
hrad2=subs(hz2,exp(i*rad));
ezplot(abs(hrad2));
grid on;
```

通过计算得到模拟低通滤波器的系统函数为：

```
.2457/(1.*i*rad+.1395-.9834*i)/(1.*i*rad+.3369-.4073*i)/(1.*i*rad+.3369+.
4
073*i)/(1.*i*rad+.1395+.9834*i)
```

数字低通滤波器的系统函数为：

```
(-.3445+.4837e-1*i)*z/((2.648-.3718*i)*z-1.)+(.2464e-1+.1755*i)*z/((2.648
-.3718*i)*z-1.)+(.4911-.1721*i)*z/((1.418-.4969*i)*z-1.)+(-.3293e-1-.9397
e-1*i)*z/((1.418-.4969*i)*z-1.)+(-.2175+.7622e-1*i)*z/((.6281-.2201*i)*z-
1.)+(-.1459e-1-.4162e-1*i)*z/((.6281-.2201*i)*z-1.)+(.4818e-1-.6764e-2*i)
*z/((.3703-.5199e-1*i)*z-1.)+(.3445e-2+.2454e-1*i)*z/((.3703-.5199e-1*i)*
z-1.)
```

由此转换得到的素质高通滤波器的系统函数为：

```
(-.3445+.4837e-1*i)*(-1.+.1416/z)/(1/z-.1416)/((2.648-.3718*i)*(-1.+.141
6/z)/(1/z-.1416)-1.)+(.2464e-1+.1755*i)*(-1.+.1416/z)/(1/z-.1416)/((2.64
8-.3718*i)*(-1.+.1416/z)/(1/z-.1416)-1.)+(.4911-.1721*i)*(-1.+.1416/z)/(
1/z-.1416)/((1.418-.4969*i)*(-1.+.1416/z)/(1/z-.1416)-1.)+(-.3293e-1-.93
```

```
97e-1*i)*(-1.+.1416/z)/(1/z-.1416)/((1.418-.4969*i)*(-1.+.1416/z)/(1/z-.
1416)-1.)+(-.2175+.7622e-1*i)*(-1.+.1416/z)/(1/z-.1416)/((.6281-.2201*i)
*(-1.+.1416/z)/(1/z-.1416)-1.)+(-.1459e-1-.4162e-1*i)*(-1.+.1416/z)/(1/z
-.1416)/((.6281-.2201*i)*(-1.+.1416/z)/(1/z-.1416)-1.)+(.4818e-1-.6764e-
2*i)*(-1.+.1416/z)/(1/z-.1416)/((.3703-.5199e-1*i)*(-1.+.1416/z)/(1/z-.1
416)-1.)+(.3445e-2+.2454e-1*i)*(-1.+.1416/z)/(1/z-.1416)/((.3703-.5199e-
1*i)*(-1.+.1416/z)/(1/z-.1416)-1.)
```

得到的高通滤波器的频率幅度特性如图 60 所示。

第 2 种方法：由模拟的低通滤波器转换成所需的模拟滤波器，然后再把它转换成数字滤波器。这种方法的第 2 步在前面已经讨论过，它可以利用冲击响应不变法和双线性变换法实现。关键是第 1 步，即由低通模拟滤波器原型到其他形式的转换。这种转换有各种各样的方法，下面给出一种将截止频率为 1 的低通滤波器转换成另一个模拟低通滤波器、带通滤波器、高通滤波器或带阻滤波器的一组关系变换式：

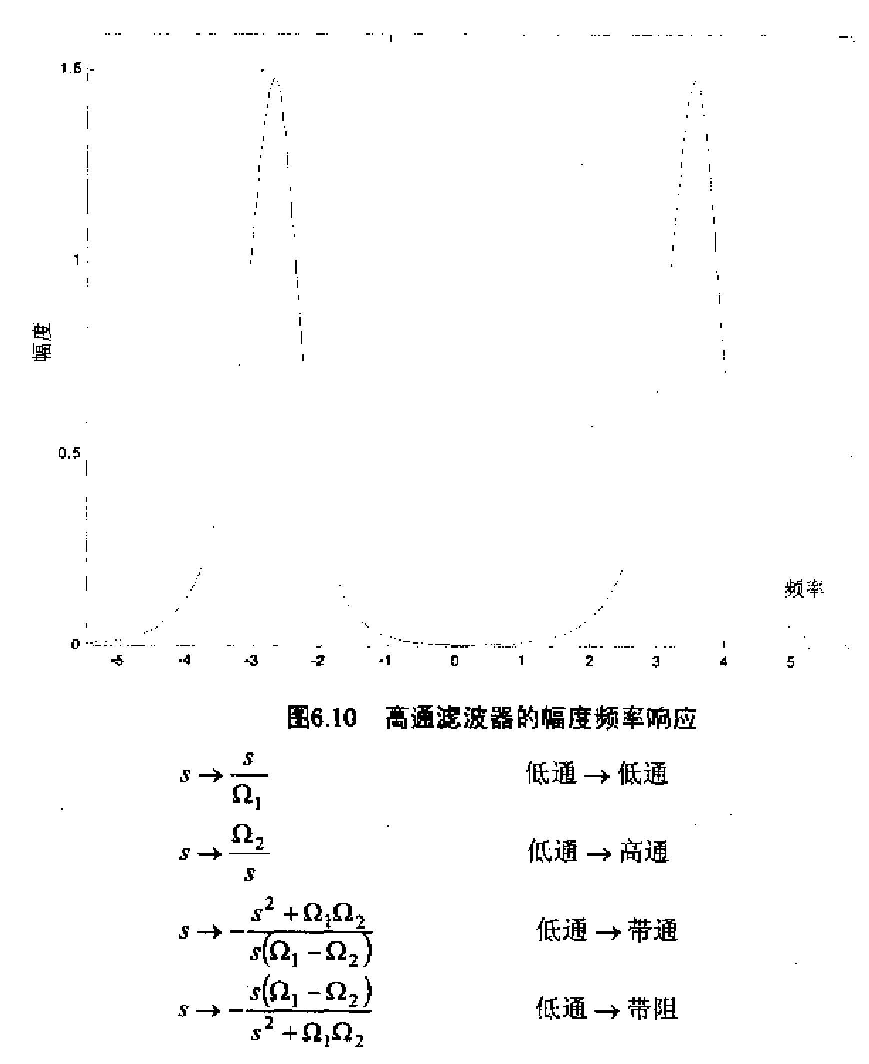


图6.10 高通滤波器的幅度频率响应

$$s \to \frac{s}{\Omega_1} \qquad \text{低通} \to \text{低通}$$

$$s \to \frac{\Omega_2}{s} \qquad \text{低通} \to \text{高通}$$

$$s \to \frac{s^2+\Omega_1\Omega_2}{s(\Omega_1-\Omega_2)} \qquad \text{低通} \to \text{带通}$$

$$s \to \frac{s(\Omega_1-\Omega_2)}{s^2+\Omega_1\Omega_2} \qquad \text{低通} \to \text{带阻}$$

上式中 Ω_1 和 Ω_2 分别表示上截止频率和下截止频率。

第 3 种方法：由模拟低通滤波器直接转换成所需的数字滤波器。这种转换方法是把 s 平面到 z 平面的映射及 z 到 Z 平面的转换统一考虑，从而得到 s 到 Z 平面的转换表达式。因此这种方法只适应于双线性变换法，而不适用于冲激响应不变法，因为后者没有确切的

变换表达式。在表 6.2 中我们给出了不同类型数字滤波器的转换关系式。

表 6.2 不同类型数字滤波器的转换关系式

滤波器类型	$s=G(Z^{-1})$	$\Omega=G(\omega)$
高通	$s=c\dfrac{1+Z^{-1}}{1-Z^{-1}}$	$\Omega=-c\cot\left(\dfrac{\omega}{2}\right)$
带通	$s=c\dfrac{Z^2-2Z^1\cos\omega_0+1}{Z^2-1}$	$\Omega=c\dfrac{\cos\omega_0-\cos\omega}{\sin\omega}$
带阻	$s=c\dfrac{Z^2-1}{Z^2-2Z^1\cos\omega_0+1}$	$\Omega=c\dfrac{\sin\omega}{\cos\omega_0-\cos\omega}$

其中表中的 c 是一个常数，ω_0是带通的中心频率。

6.5.2 直接法设计 IIR 数字滤波器

利用直接法设计 IIR 数字滤波器可以在时域或频域上进行。在频域进行直接设计，目前有 3 种方法：z 平面的简单零极点法、幅度平方函数法、频域优化设计法。在时域进行直接设计，目前有两种方法：博德逼近化和最小平方逆优化法。由于篇幅的限制，本章只讨论简单零极点法和幅度平方函数法。

1. z平面的简单零极点法

数字滤波器的系统函数 H(z)可以表示为：

$$H(z)=\frac{\prod_{k=1}^{M}\left(1-c_r z^{-1}\right)}{\prod_{k=1}^{N}\left(1-d_r z^{-1}\right)}$$

式中的 c_2、d_2 分别表示滤波器的零点和极点。由前面的知识我们知道，零点和极点的位置完全决定了滤波器的幅度函数和相位函数，可以用选择零点和极点的位置，按照对滤波器幅度和相位的要求设计所需的滤波器。这种方法通常只能用于设计简单的 IIR 数字滤波器，但是它的物理概念比较清晰，所以在此首先进行讨论。

- 低通滤波器的设计

  低通(或高阻)滤波器，即使其性能在 ω＝π 处的传输系数为零，相当于在 z=－1 前一个零点，在 z=a 处有一个极点，a 为小于 1 的正实数，a 越大，滤波器的带宽越窄，所以最简单的低通滤波器的系统函数可以表示为：

$$H(z)=\frac{1+z^{-1}}{1-az^{-1}}$$

式中的 a 可以根据带宽的要求决定，如图 6.11 所示，可以得到：

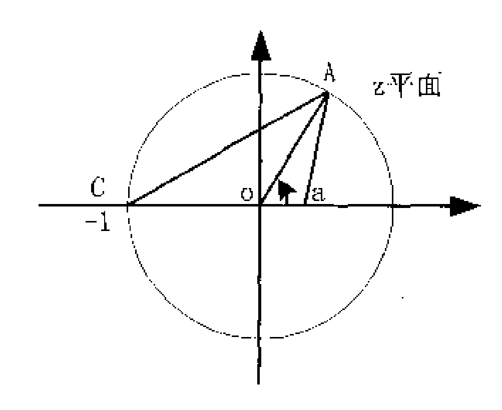


图6.12 用Z平面简单零、极点设计简单的滤波器

$$|H(j\omega)|^2=\frac{|CA|^2}{|Aa|^2}$$
$$=\frac{CO^2+AO^2-2CO\bullet AO\cos(\pi-\omega)}{AO^2+a^2-2a\bullet AO\cos(\omega)}$$
$$=\frac{2+2\cos\omega}{1-2a\cos\omega+a^2}$$

当ω=0时，函数达到最大：

$$|H(j0)|^2=\max|H(j\omega)|^2=\frac{4}{(1-a)^2}$$

如果给定带宽ω_1，取：

$$|H(j\omega_1)|^2=\frac{1}{2}|H(j0)|^2$$
$$=\frac{1}{2}\frac{4}{(1-a)^2}$$

可以解得：

$$a=\frac{1\pm\sin\omega_1}{\cos\omega_1}$$

因为要求设计一个因果稳定的系统，所以a的绝对值小于1，得：

$$a=\frac{1-\sin\omega_1}{\cos\omega_1}$$

经过归一化处理，得到归一化的低通滤波器的系统函数为：

$$H(z)=\frac{1-a}{2}\frac{1+z^{-1}}{1-az^{-1}}$$

例，设计一个带宽为2kHz的数字低通滤波器，若抽样频率为8倍带宽，利用z平面的简单零极点法设计一个低通滤波器。

我们可以利用MATLAB编写一个简单小程序得到系统函数，并且可以分析这个滤波器的性能。程序清单如下：

```
syms rad z hz hrad angle;
    passrad=pi/4;
```

```
a=(1-sin(passrad))/cos(passrad);
hz=(1+1/z)/(1-a/z)*(1-a)/2;
hrad=subs(hz,exp(i*rad));
subplot(2,1,1);
ezplot(abs(hrad),[-2*pi,2*pi]);
grid on;
subplot(2,1,2);
angle=atan(imag(hrad)/real(hrad));
ezplot(angle);
grid on;
```

得到的滤波器的系统函数为：

```
H(z)=.29289*(1.+1/z)/(1.-.41421/z)
```

从而得到滤波器的幅度特性$|H(j\omega)|$和相位特性$\arg(H(j\omega))$，它们的特性如图 62 所示。

- 高通滤波器的设计

  将低通滤波器的零点和极点的位置互换，就可以得到高通数字滤波器的系统函数为：

$$H(z)=\frac{1-a}{2}\frac{1-z^{-1}}{1+az^{-1}}$$

$$a=\frac{\cos\omega_2}{1+\sin\omega_2}$$

  其中 ω 为高通滤波器的截止频率。

- 带通滤波器的设计

  如果把低通滤波器和高通滤波器组合起来，即可以得到带通滤波器，它的系统函数可以表示为两种滤波器系统函数的乘积。

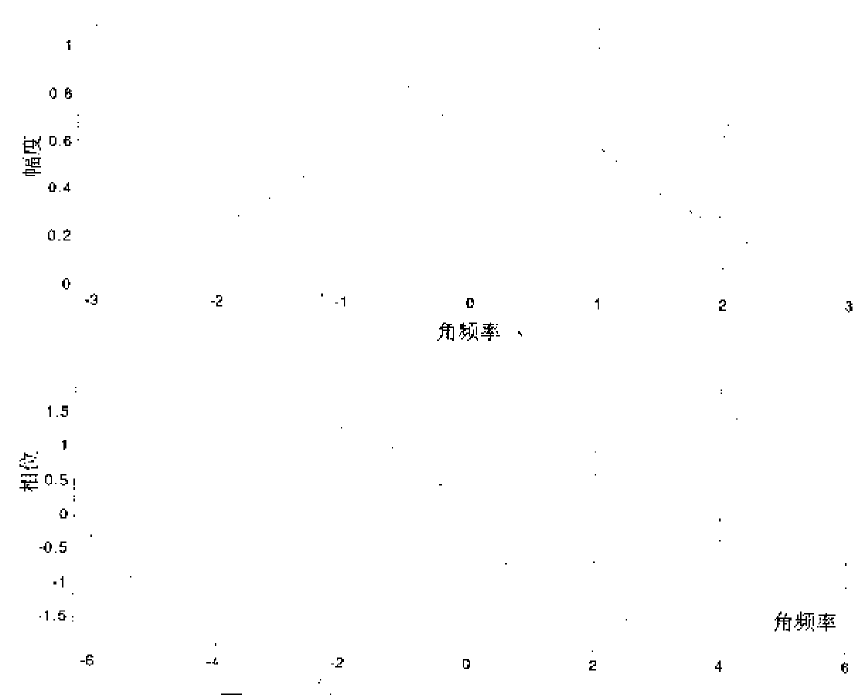


图6.12　低通滤波器的幅度和相位特性

$$H(z)=\frac{1-z^{-1}}{1+a_2z^{-1}}\bullet\frac{1+z^{-1}}{1-a_1z^{-1}}$$

其中：

$$a_2=\frac{\cos\omega_2}{1+\sin\omega_2}$$

$$a_1=\frac{\cos\omega_1}{1+\sin\omega_1}$$

通带的中心频率为：

$$\omega_0=\arccos\left(\frac{a_1-a_2}{1-a_1a_2}\right)$$

由此可见，带通滤波器可以通过一个二阶系统来实现。

2. 幅度平方函数法

模仿模拟滤波器的幅度平方函数，可以直接利用数字滤波器的幅度平方函数进行设计。IIR 数字滤波器的系统函数为：

$$H(z)=\frac{\sum_{i=0}^{M}a_iz^{-i}}{\sum_{i=0}^{N}b_iz^{-i}}$$

当系统函数的系数 a_i、b_i 都是实数时，它的幅度平方函数可以表示为：

$$|H(j\omega)|^2=\left[H(z)H(z^{-1})\right]_{z=e^{j\omega}}$$

也可以表示为：

$$|H(j\omega)|^2=\frac{\sum_{i=0}^{M}a_ie^{-j\omega i}\sum_{i=0}^{M}a_ie^{j\omega i}}{\sum_{i=0}^{N}b_ie^{-j\omega i}\sum_{i=0}^{N}b_ie^{j\omega i}}$$

通过三角恒等变换得到：

$$|H(j\omega)|^2=\frac{\sum_{i=0}^{M}c_i\cos^2\left(i\omega/2\right)}{\sum_{i=0}^{N}f_i\cos^2\left(i\omega/2\right)}$$

如果模仿模拟滤波器的型式，上式可以写成：

$$|H(j\omega)|^2=\frac{1}{1+A_N^2(\omega)}$$

因此，如果能够找到合适的 $A_N^2(\omega)$，就可以建立一套类似于设计模拟滤波器的办法直接设计数字滤波器，但是这个方法有两个困难：第一，必须寻找出一个适当的有理三角函数多项式，用它求出数字滤波器；第二，幅度平方函数必须是可以因式分解的，由此找出零点和极点，但是因式分解并非任何情况下都是可以的。因此尽管一些文献给出了一些特殊情况下的结果，但是还没有一个一般的方法提供设计，因此幅度平方函数并不是一个实用的设计方法。

6.6 利用 MATLAB 直接设计 IIR 数字滤波器

通常可以利用 MATLAB 很方便的直接设计出所需的滤波器，总体上来说，MATLAB 中一共有 4 种设计 IIR 滤波器的方法，它们是模拟原型法、直接法、参数模型法和通用巴特沃斯设计法，我们在这一小节中将介绍和分析 MATLAB 中模拟原型法中的 butterworth、cheby1、cheby2、ellip 等方法以及 yulewalk 方法。

6.6.1 巴特沃斯数字滤波器设计

在 MATLAB 中，可以利用 butter 函数直接设计各种形式的数字滤波器(也可以设计模拟滤波器)，它的语法为：

```
[b,a] = butter(n,Wn)
[b,a] = butter(n,Wn,'ftype')
[z,p,k] = butter(...)
[A,B,C,D] = butter(...)
```

butter 函数可以设计低通、高通、带通和带阻各种形式的滤波器，在前面的模拟滤波器设计中，我们知道 butterworth 滤波器的幅度特性是通带平坦、阻带单调下降。但是 Chebyshev 滤波器和椭圆滤波器在满足相同的设计要求下，滤波器的阶数一般比 Butterworth 低。

利用[b,a] = butter(n,Wn)方式可以设计一个阶数为 n、截止频率为 Wn 的低通滤波器。它的返回值 a 和 b 为系统函数的分子和分母的系数。系统函数可以表示为：

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}$$

截止频率 wn 是指滤波器的半功率点，它的取值范围在 0 到 1 之间，其中 1 对应为采样频率的一半。如果 Wn 是一个含有两个元素的向量[w1 w2]，则返回的[a，b]所构成的滤波器是阶数为 2n 的带通滤波器，通带范围为 $w1 < W < w2$。

利用[b,a] = butter(n,Wn,'ftype')方式可以设计高通、带阻滤波器，其中参数 ftype 的形式确定了滤波器的形式，当它为'high'时，得到的滤波器为 n 阶的、截止频率为 Wn 的高通滤波器；当它为'stop'时，得到的滤波器是阶数为 2n、阻带范围为 $w1 < W < w2$ 的带阻滤波器。

利用返回值数目的不同可以得到滤波器的其他表达形式，例如，如果返回值的数目是 3 个，得到的返回值分别是滤波器的零点、极点和增益，函数的参数和前面的形式相同，具体的语法形式为：

```
[z,p,k] = butter(n,Wn) or
[z,p,k] = butter(n,Wn,'ftype')
```

如果返回值的数目为 4 个，就可以得到滤波器的状态空间的表达形式，具体的语法形式如下：

```
[A,B,C,D] = butter(n,Wn) or
[A,B,C,D] = butter(n,Wn,'ftype')
```

可以利用返回值 A, B, C 和 D 构造滤波器的状态方程：

$$x(n+1) = Ax(n) + Bu(n)$$
$$y(n) = Cx(n) + Du(n)$$

其中 u 是输入信号，x 是状态变量，y 是输出信号。

例，对于采样频率为 1000 Hz 的采样信号，设计一个阶数为 9 阶、截止频率为 300 Hz 的高通 butterworth 数字滤波器。

```
[b,a] = butter(9,300/500,'high');
```

得到滤波器分子和分母的系数，从而可以得到滤波器的频率响应，如图 6.13 所示。

```
freqz(b,a,128,1000)
```

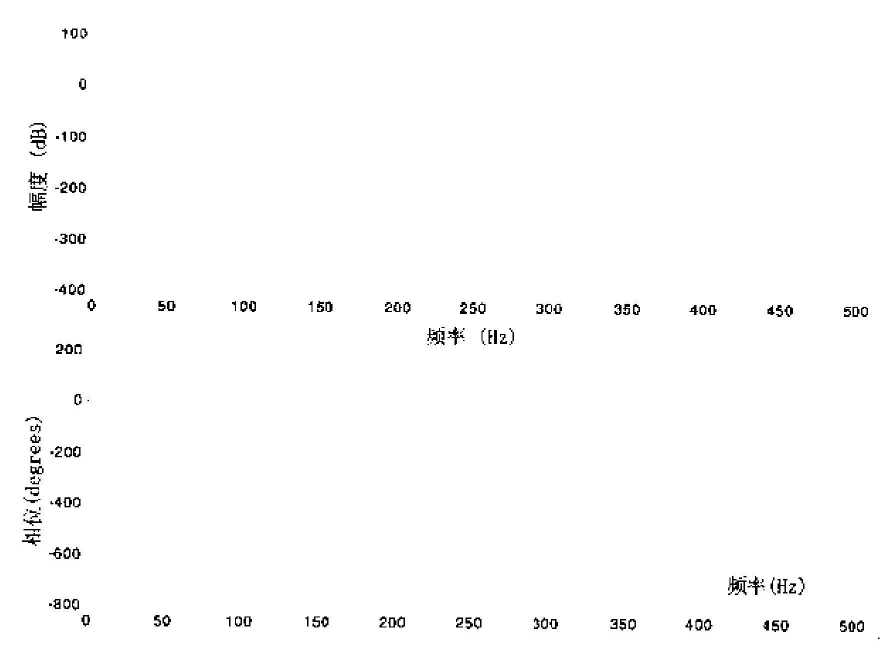


图6. 13　高通滤波器的频率特性

例，设计一个 10 阶的带通 butterworth 滤波器，它的通带范围是 100 到 200 Hz，并画出它的冲激响应。程序的清单和结果如下：

```
n = 5;
Wn = [100 200]/500;
[b,a] = butter(n,Wn);
[y,t] = impz(b,a,101);
stem(t,y)
```

它的冲激响应如图 6.14 所示。

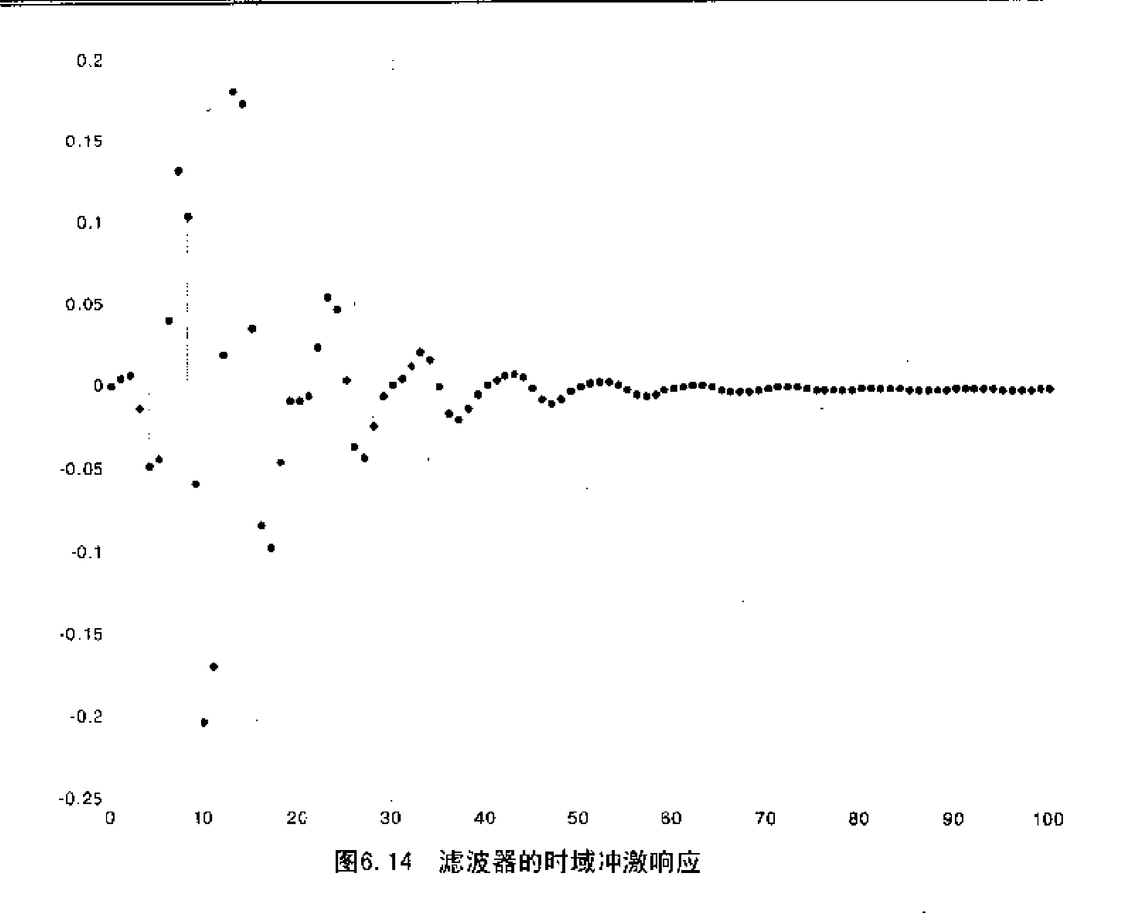


图6.14 滤波器的时域冲激响应

6.6.2 椭圆法数字滤波器设计

在 MATLAB 中，可以利用 ellip 函数直接设计各种形式的数字滤波器(也可以设计模拟滤波器)，它的语法为：

```
[b,a] = ellip(n,Rp,Rs,Wn)
[b,a] = ellip(n,Rp,Rs,Wn,'ftype')
[z,p,k] = ellip(...)
[A,B,C,D] = ellip(...)
```

实际上，ellip 函数是采用椭圆法设计出低通的模拟滤波器，然后采用变换的方法得到数字的高通、低通、带通和带阻滤波器，所以称之为 ellip 函数。在模拟滤波器的设计中，椭圆滤波器的设计是几种滤波器设计方法中最为复杂的一种设计方法，但是它设计出的滤波器的阶数最小，同时它对参数的量化灵敏度最敏感。

可以利用[b,a] = ellip(n,Rp,Rs,Wn)方式设计出阶数为 n、截止频率为 Wn、通带波纹最大衰减为 Rp、阻带波纹最小衰减为 Rs 的数字低通滤波器，它的返回值 a、b 分别是阶数为 n + 1 的向量，表示数字低通滤波器的系统函数的分子和分母的多项式系数。滤波器的系数可以表示为：

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}$$

函数的截止频率 Wn 是指通带的边缘，在那儿滤波器的幅度响应为-Rp dB，Wn 的取值

范围为 0 到 1，其中 1 表示采样频率的一半。滤波器的过渡带宽取决于 3 个参数 n、Rp、Rs，当 n 不变时，Rp 越小，Rs 越大，过渡带就越宽。

如果 Wn 是一个含有两个元素的向量[w1 w2]，则 ellip 函数返回值是阶数为 2n 的带通滤波器的系统函数有理多项式的系数，滤波器的通带范围是 $w1 < W < w2$。

可以利用[b,a] = ellip(n,Rp,Rs,Wn,'ftype')方式设计高通和带阻滤波器，其中参数 ftype 的形式确定了滤波器的形式，当它为'high'时，得到的滤波器为 n 阶的、截止频率为 Wn 的高通滤波器；当它为'stop'时，得到的滤波器是阶数为 2n、阻带范围为 $w1 < W < w2$ 的带阻滤波器。

利用返回值数目的不同可以得到滤波器的其他表达形式，例如，如果返回值的数目是 3 个，得到的返回值分别是滤波器的零点、极点和增益，函数的参数和前面的形式相同，具体的语法形式为：

```
[z,p,k] = ellip(n,Rp,Rs,Wn) or
[z,p,k] = ellip(n,Rp,Rs,Wn,'ftype')
```

如果返回值的数目为 4 个，就可以得到滤波器的状态空间的表达形式，具体的语法形式如下：

```
[A,B,C,D] = ellip(n,Rp,Rs,Wn) or
[A,B,C,D] = ellip(n,Rp,Rs,Wn,'ftype')
```

可以利用返回值 A、B、C 和 D 构造滤波器的状态方程：

$$x(n+1) = Ax(n) + Bu(n)$$
$$y(n) = Cx(n) + Du(n)$$

其中 u 是输入信号，x 是状态变量，y 是输出信号。

例，对于采样频率为 1000 Hz 的采样信号，设计一个阶数为 6 阶、截止频率为 300 Hz 的低通 ellip 数字滤波器，其中滤波器在通带的波纹为 3 dB，在阻带的波纹为 50 dB。我们利用 ellip 函数设计：

```
[b,a] = ellip(6,3,50,300/500);
```

从而可以得到滤波器的频率特性：

```
freqz(b,a,512,1000)
title('n=6 Lowpass Elliptic Filter')
```

滤波器的频率响应如图 6.15 所示。

例，设计一个 20 阶的带通滤波器，它的通带范围为 100 到 200Hz，其中信号的采样频率为 1000Hz，Rp = 0.5；Rs = 20，设计出这个滤波器并画出它的冲激响应。程序的清单和结果如下：

```
n = 10;
Rp = 0.5;
Rs = 20;
Wn = [100 200]/500;
[b,a] = ellip(n,Rp,Rs,Wn);
[y,t] = impz(b,a,101);
```

```
    stem(t,y)
  title('图 6.16 椭圆法设计带通滤波器的冲激响应')
```

滤波器的冲激响应如图6.16所示。

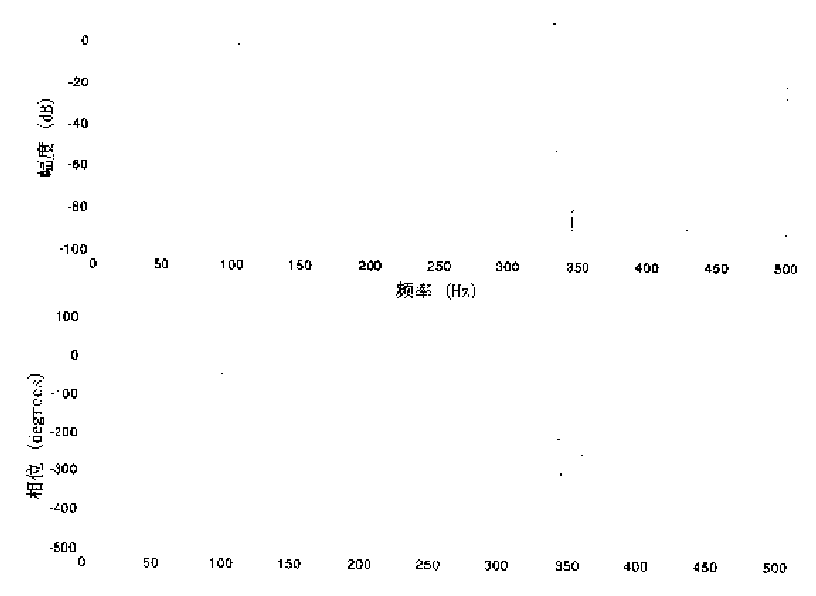


图6.15　椭圆法设计的低通滤波器的频率响应

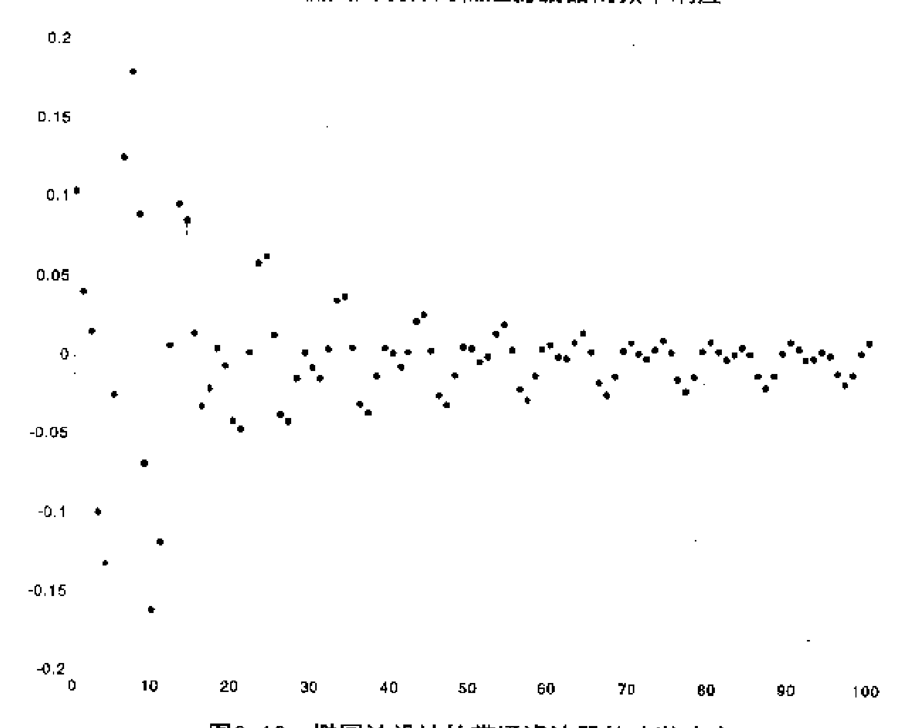


图6.16　椭圆法设计的带通滤波器的冲激响应

6.6.3　切比雪夫 1 法数字滤波器设计

在 MATLAB 中，可以利用 cheby1 函数直接设计各种形式的数字滤波器(也可以设计模拟滤波器)，它的语法为：

```
[b,a] = cheby1(n,Rp,Wn)
[b,a] = cheby1(n,Rp,Wn,'ftype')
[z,p,k] = cheby1(...)
[A,B,C,D] = cheby1(...)
```

实际上，cheby1函数是采用切比雪夫1法设计出低通的模拟滤波器，然后采用变换的方法得到数字的高通、低通、带通和带阻滤波器，所以称之为cheby1函数。在模拟滤波器的设计中，Chebyshev1滤波器在通带是等波纹的，而在阻带是单调下降的；Chebyshev2滤波器则相反。

可以利用[b,a] = cheby1(n,Rp,Wn)方式设计出阶数为n、截止频率为Wn、通带波纹最大衰减为Rp的数字低通滤波器，它的返回值a、b分别是阶数为n+1的向量，表示数字低通滤波器的系统函数的分子和分母的多项式系数。滤波器的系数可以表示为：

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}$$

函数的截止频率Wn是指通带的边缘，在那儿滤波器的幅度响应为−Rp dB，Wn的取值范围为0到1，其中1表示采样频率的一半。越小的通带波纹会导致越大的过渡带宽。

如果Wn是一个含有两个元素的向量[w1 w2]，则cheby1函数返回值是阶数为2n的带通滤波器的系统函数有理多项式的系数，滤波器的通带范围是w1 < W< w2。

可以利用[b,a] = cheby1 (n,Rp，Wn,'ftype')方式设计高通和带阻滤波器，其中参数ftype的形式确定了滤波器的形式，当它为'high'时得到的滤波器为n阶的、截止频率为Wn的高通滤波器；当它为'stop'时，得到的滤波器是阶数为2n、阻带范围为w1 < W< w2的带阻滤波器。

利用返回值数目的不同可以得到滤波器的其他表达形式，例如，如果返回值的数目是3个，得到的返回值分别是滤波器的零点、极点和增益，函数的参数和前面的形式相同，具体的语法形式为：

```
[z,p,k] = cheby1(n,Rp,Wn) or
[z,p,k] = cheby1(n,Rp,Wn,'ftype')
```

如果返回值的数目为4个，就可以得到滤波器的状态空间的表达形式，具体的语法形式如下：

```
[A,B,C,D] = cheby1(n,Rp,Wn) or
[A,B,C,D] = cheby1(n,Rp,Wn,'ftype')
```

可以利用返回值A、B、C和D构造滤波器的状态方程：

$$x(n+1)=Ax(n)+Bu(n)$$
$$y(n)=Cx(n)+Du(n)$$

其中u是输入信号，x是状态变量，y是输出信号。

例，对于采样频率为1000 Hz的采样信号，设计一个阶数为9阶、截止频率为300 Hz的低通cheby1数字滤波器，其中滤波器在通带的波纹为0.5 dB。我们利用cheby1函数设计：

```
[b,a] = cheby1(9,0.5,300/500);
```

从而可以得到滤波器的频率特性：

```
freqz(b,a,512,1000)
```

滤波器的频率响应如图 6.17 所示。

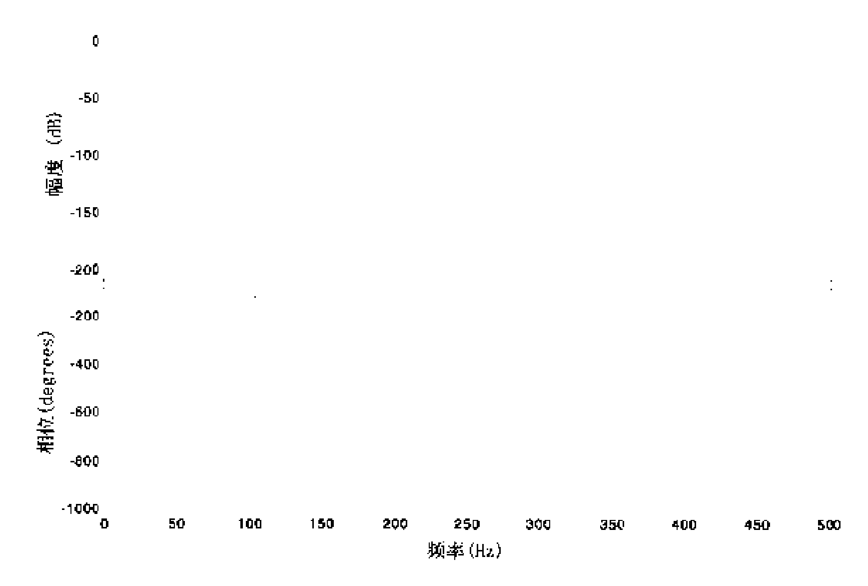


图6.17 用cheby1函数设计的低通滤波器的频率响应

例如设计一个 10 阶的带通 Chebyshevl 滤波器，它的通带范围是 100 到 200 Hz ，其中信号的采样频率为 1000Hz，n = 10; Rp = 0.5。设计出滤波器并画出系统函数的冲激响应。程序清单和结果如下：

```
n = 10;
Rp = 0.5;
Wn = [100 200]/500;
[b,a] = cheby1(n,Rp,Wn);
[y,t] = impz(b,a,101);
stem(t,y)
```

滤波器的冲激响应如图 6.18 所示。

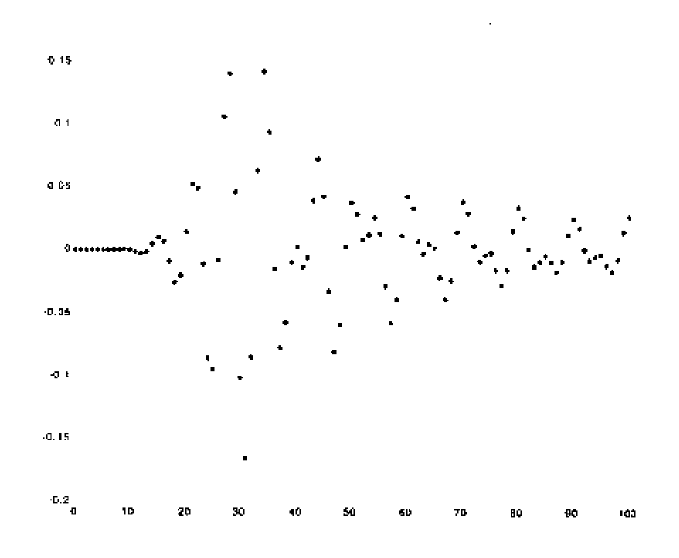


图6.18 滤波器的冲激响应

6.6.4　切比雪夫 2 法数字滤波器设计

利用 MATLAB 中的 cheby2 设计各种形式的滤波器的方法和利用 cheby1 设计滤波器的方法相同，只是 cheby2 设计出的滤波器在阻带是等波纹的，在通带是单调的；而 cheby1 则恰好相反。我们在此不作详细的讨论，只引用了一个例子。

例，对于采样频率为 1000 Hz 的采样信号，设计一个阶数为 9 阶、截止频率为 300 Hz 的低通 Chebyshev II 数字滤波器，其中滤波器在阻带的波纹为 20dB。我们利用 cheby2 函数设计：

```
[b,a] = cheby2(9,20,300/500);
```

从而可以得到滤波器的频率特性：

```
freqz(b,a,512,1000)
```

滤波器的频率响应如图 6.19 所示。

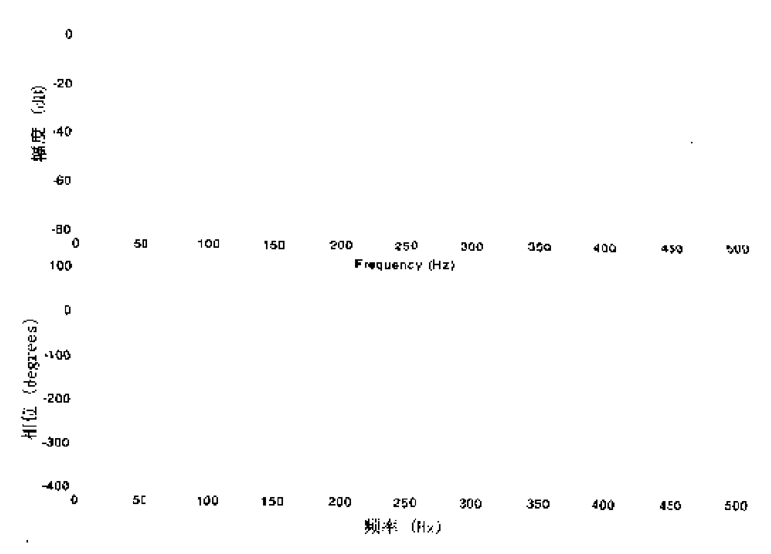


图6.19　利用chely2设计的低通滤波器的频率响应

6.6.5　yulewalk 法数字滤波器设计

yulewalk 法设计数字滤波器实际上是一种递归的数字滤波器设计，只能利用这个函数进行数字滤波器的设计，而不能进行模拟滤波器的设计。它是一种在频域中采用了最小均方法进行设计滤波器的方法，在 MATLAB 中它的语法形式是：

```
[b,a] = yulewalk(n,f,m)
```

yulewalk(n,f,m) 返回向量 b 和 a ，它们分别是阶数为 n + 1 的向量，表示数字低通滤波器的系统函数的分子和分母的多项式系数。

$$H(z)=\frac{B(z)}{A(z)}=\frac{b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}}{1+a(2)z^{-1}+\cdots+a(n+1)z^{-n}}$$

滤波器幅频特性逼近于 f 和 m 确定的理想滤波器的幅频特性。其中 f 是一个向量，它的每一元素都是 0 到 1 的数，它们表示频率，1 表示采样频率的一半，向量中的元素必须是递增的，第 1 个元素必须是 0，最后一个元素必须是 1。m 是频率 f 处的幅度响应，它也是一个向量，并且向量的长度和 f 相同。

当确定了理想滤波器的幅度频率响应后，为了避免从通带到阻带的过渡陡峭，应该对过渡带进行多次试验，以便得到最好的滤波器设计。

例，设计一个 8 阶的低通数字滤波器，对应的理想滤波器的截止频率为 300Hz，信号的采样频率为 1000 Hz。我们利用 yulewalk 函数设计，程序的清单如下：

```
f = [0 0.6 0.6 1];
m = [1 1 0 0];
[b,a] = yulewalk(8,f,m);
[h,w] = freqz(b,a,128);
plot(f,m,w/pi,abs(h),'--')
```

得到的滤波器的频率特性如图 6.20 所示。

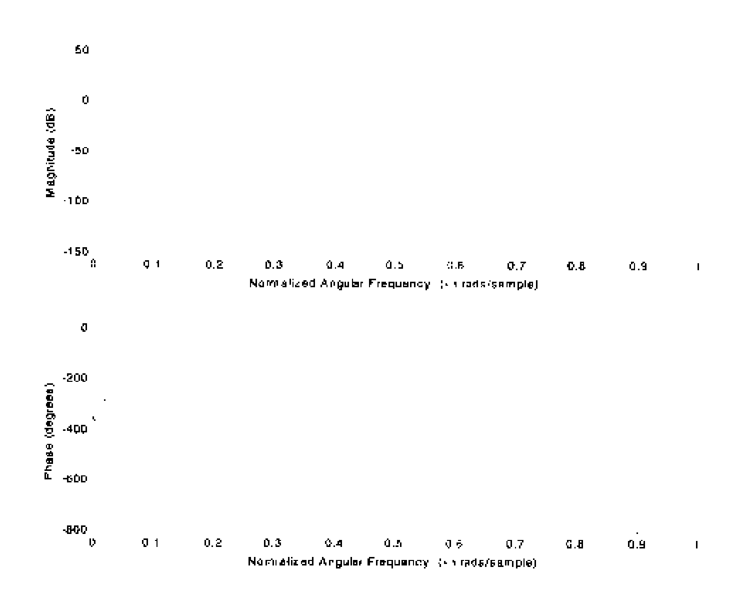


图6. 20　滤波器的频率特性

将它和理想的低通滤波器相比较如图 6.21 所示。

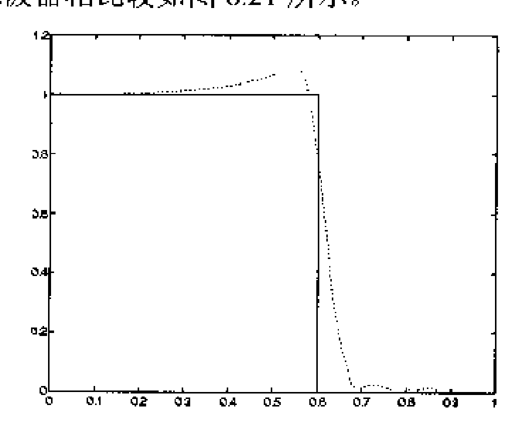


图6. 21　与理想的低通滤波器比较图

第 7 章 基于 MATLAB 的 FIR 数字滤波器设计

在前一章中讨论了 IIR 数字滤波器的设计以及基于 MATLAB 的 IIR 数字滤波器的设计，由于设计 IIR 数字滤波器能够保留一些模拟滤波器的优良特性，因此得到了广泛的应用。但是这些特性的获得是以牺牲线性相位频率特性为代价的。换句话说，用巴特沃斯、切比雪夫和椭圆法设计的数字滤波器逼近理想的滤波器的幅度频率特性，得到的滤波器的特性往往是非线性的。在很多实际的电子系统中，既要求有良好的幅度频率特性，又要求有线性相位特性。在这方面，有限长单位脉冲响应(FIR)数字滤波器具有优良的特点。它可以在设计任意幅度频率特性的滤波器的同时，保证精确、严格的线性相位特性。FIR 数字滤波器的单位脉冲响应 h(n)是有限长的，可以用一个因果系统来实现，因而 FIR 数字滤波器可以做成既是因果的又是稳定的系统。FIR 的这些特性使得它的应用越来越广泛。

设计 FIR 数字滤波器的方法有窗函数法、频率取样法和切比雪夫逼近法等，在这一章中将分别进行讨论。同时还将介绍 MATLAB 设计 FIR 数字滤波器的一些有用的函数以及 MATLAB 设计 FIR 数字滤波器的工具。

7.1 窗函数及 MATLAB 的实现和分析

窗函数在设计 FIR 数字滤波器中有很重要的作用，正确地选择窗函数可以提高所设计的数字滤波器的性能，或者在满足设计要求的情况下，减小 FIR 数字滤波器的阶数。因此，必须对各种窗函数有相应的了解。在本节中，将讨论常见的 8 种窗函数的形式、特性以及如何利用 MATLAB 实现它们。这其中常见的窗函数是：矩形窗(Rectangular window)、三角窗(Triangular window)、布拉克曼窗(Bartlett window)、汉宁窗(Hanning window)、海明窗(Hamming window)、凯塞窗(Kaiser window)、巴特里特窗(Bartlett window)、切比雪夫窗(Chebyshev window)。下面将对他们逐一进行讨论。

7.1.1 矩形窗

矩形窗函数的时域形式可以表示为：

$$w(n)=R_N(n)=\begin{cases}1 & 0\leqslant n\leqslant n-1\\ 0 & \text{其他的 n}\end{cases}$$

它的频域特性为：

$$W_R\left(e^{j\omega}\right)=e^{-j\omega\left(\frac{N-1}{2}\right)}\frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}$$

它的幅度频率特性和时域特性如图 7.1 所示(选择窗函数的长度为 51)。

在 MATLAB 中可以利用 w = boxcar(n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，它的元素由窗函数的值组成。其实'w = boxcar(n)'等价于'w = ones(n,1)'。

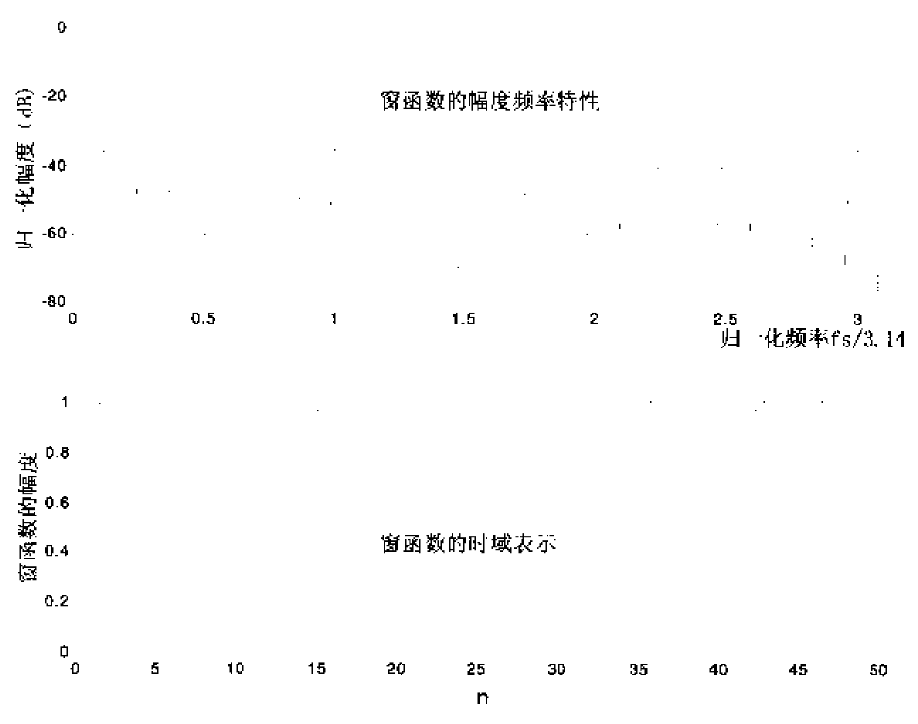


图7.1　矩形窗函数的特性

7.1.2　三角窗

三角窗函数的时域形式可以表示为：

对于 n 为奇数：

$$w(k)=\begin{cases}\dfrac{2k}{n+1} & 1\leqslant k\leqslant\dfrac{n+1}{2}\\ \dfrac{2(n-k+1)}{n+1} & \dfrac{n+1}{2}\leqslant k\leqslant n\end{cases}$$

对于 n 为偶数：

$$w(k)=\begin{cases}\dfrac{2k-1}{n} & 1\leqslant k\leqslant\dfrac{n}{2}\\ \dfrac{2(n-k+1)}{n} & \dfrac{n}{2}+1\leqslant k\leqslant n\end{cases}$$

它的频域特性为：

$$W_R\left(e^{j\omega}\right)=e^{-j\omega\left(\frac{N-1}{2}\right)}\frac{2}{\mathrm{N}-1}\left\{\frac{\sin\left(\frac{\omega(N-1)}{4}\right)}{\sin\left(\frac{\omega}{2}\right)}\right\}^2$$

它的幅度频率特性和时域特性如图 7.2 所示(选择窗函数的长度为 51)。

三角窗函数的主瓣宽度为 8π/N，比矩形窗函数的主瓣宽度增加一倍，但是它的旁瓣却小得多。在 MATLAB 中可以利用 w = triang(n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，它的元素由窗函数的系数组成。三角窗函数和巴特里特窗函数十分相似，巴特里特函数在开始和结束时通常为零，而三角函数的第 1 个系数和最后一个系数通常不为零。当窗函数的长度 N 为偶数时，triang(N-2) a 等价于 bartlett(N)。

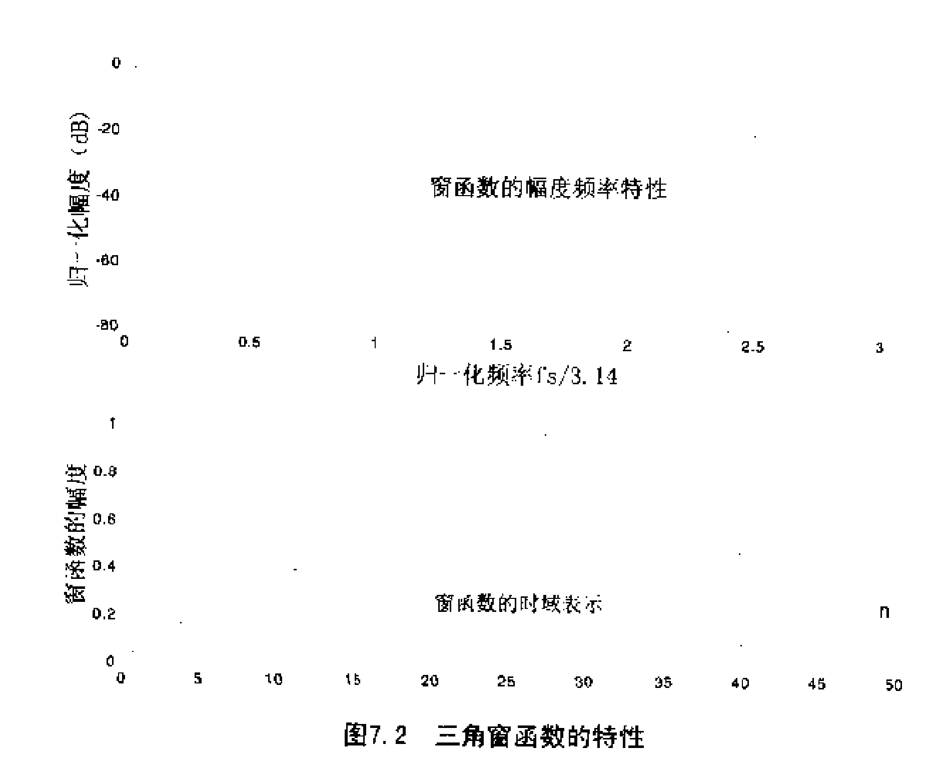


图7.2　三角窗函数的特性

7.1.3　汉宁窗

汉宁窗函数是余弦平方函数，又称之为升余弦函数，它的时域形式可以表示为：

$$w(k)=0.5\left(1-\cos(2\pi\frac{k}{n+1})\right)\qquad k=1,2,\cdots,n$$

它的频率幅度特性函数为：

$$W(\omega)=\left\{0.5W_R(\omega)+0.25\left[W_R\left(\omega-\frac{2\pi}{N-1}\right)+W_R\left(\omega+\frac{2\pi}{N-1}\right)\right]\right\}e^{-j\left(\frac{N-1}{2}\right)\omega}$$

其中 $W_R(\omega)$为矩形窗函数的幅度频率特性函数，汉宁窗函数的幅度频率特性和时域特性如图 7.3 所示(选择窗函数的长度为 51)。

汉宁窗函数的最大旁瓣值比主瓣值低 31dB，但是主瓣宽度比矩形窗函数的主瓣宽度增

加了1倍，为 8π /N。在 MATLAB 中可以利用 w = hanning (n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，包含了窗函数的 n 个系数。

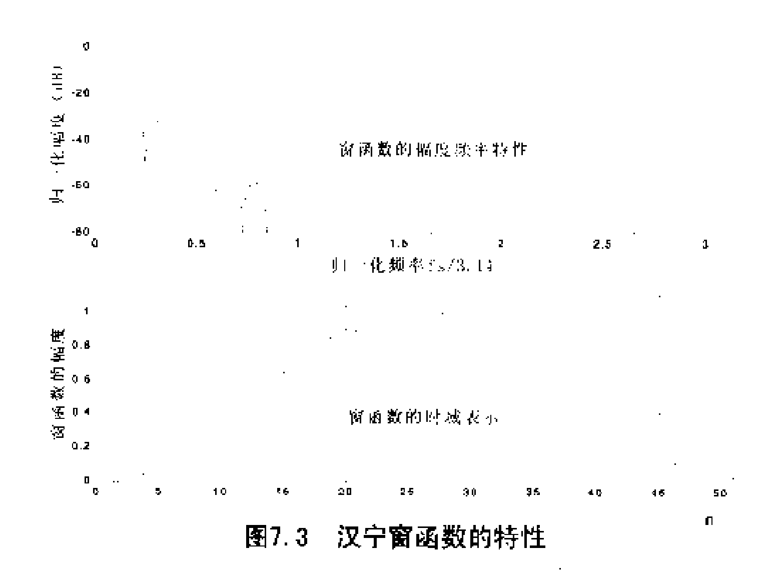


图7.3 汉宁窗函数的特性

7.1.4 海明窗

海明窗函数是一种改进的升余弦函数，定义为：

$$w(n)=0.54-0.46\cos\left(\frac{2\pi n}{N-1}\right)\qquad 0\leqslant n\leqslant N-1$$

它幅度频率特性为：

$$W(\omega)=0.54W_R(\omega)+0.23\left[W_R\left(\omega-\frac{2\pi}{N-1}\right)+W_R\left(\omega+\frac{2\pi}{N-1}\right)\right]$$

它的幅度频率特性和时域特性如图 7.4 所示(选择窗函数的长度为 51)。

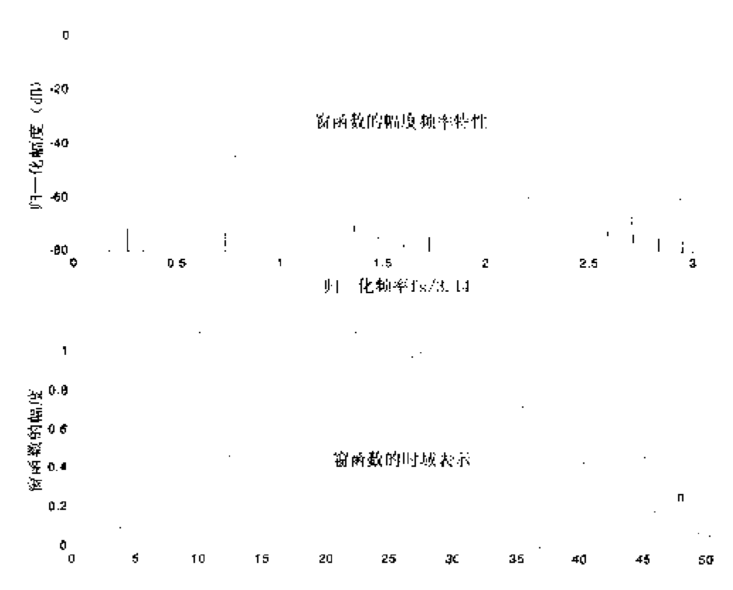


图7.4 海明窗函数的频率特性和时域特性

和汉宁窗函数相比，海明窗函数的主瓣宽度和汉宁窗相同，但是它的旁瓣又被进一步压低，其最大旁瓣值比主瓣值低 41dB。在 MATLAB 中可以利用 w = hamming (n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，包含了窗函数的 n 个系数。

7.1.5　布拉克曼窗

为了进一步抑制旁瓣，对升余弦函数再加上一个二次谐波的余弦分量，便得到了布拉克曼窗函数，也称之为二阶升余弦窗函数。它的时域形式可以表示为：

$$w(k) = 0.42 - 0.5\cos(2\pi\frac{k-1}{n-1}) + 0.08\cos(4\pi\frac{k-1}{n-1}) \qquad k = 1,2,\cdots,n$$

其幅度频率特性为：

$$W(\omega) = 0.42W_R(\omega) + 0.25\left[W_R\left(\omega - \frac{2\pi}{N-1}\right) + W_R\left(\omega + \frac{2\pi}{N-1}\right)\right] + 0.04\left[W_R\left(\omega - \frac{4\pi}{N-1}\right) + W_R\left(\omega + \frac{4\pi}{N-1}\right)\right]$$

其中 $W_R(\omega)$为矩形窗函数的幅度频率特性函数，布拉克曼窗函数的幅度频率特性和时域特性如图 7.5 所示(选择窗函数的长度为 51)。

在 MATLAB 中可以利用 w = blackman (n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，包含了窗函数的 n 个系数。布拉克曼窗函数的主瓣的宽度是矩形窗函数的主瓣宽度的 3 倍。为 12π /N，它的最大旁瓣值比主瓣值低 57dB。

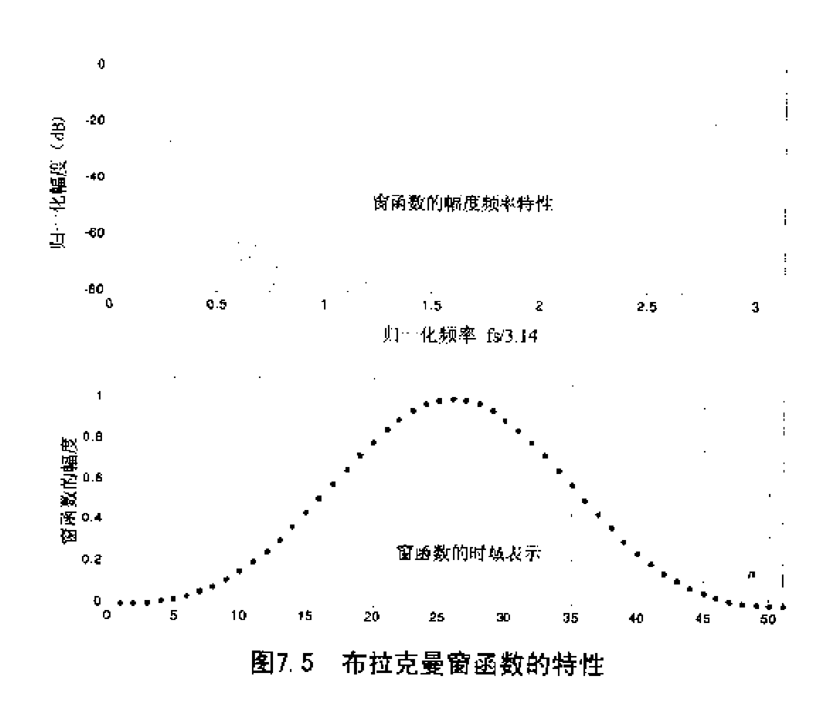


图7.5　布拉克曼窗函数的特性

7.1.6　切比雪夫窗

在第 6 章中，我们讨论了切比雪夫 II 法设计低通滤波器，它的阻带是波纹的，而在

MATLAB 中可以利用 w = chebwin(N,R)方式设计出 N 阶的切比雪夫Ⅱ窗函数，窗函数的主瓣值比旁瓣值高 RdB，且瓣旁是等波纹的。

图 7.6 显示了切比雪夫窗函数的幅度频率特性和它的时域表示(其中窗函数的长度为 51，R 为 50dB)。

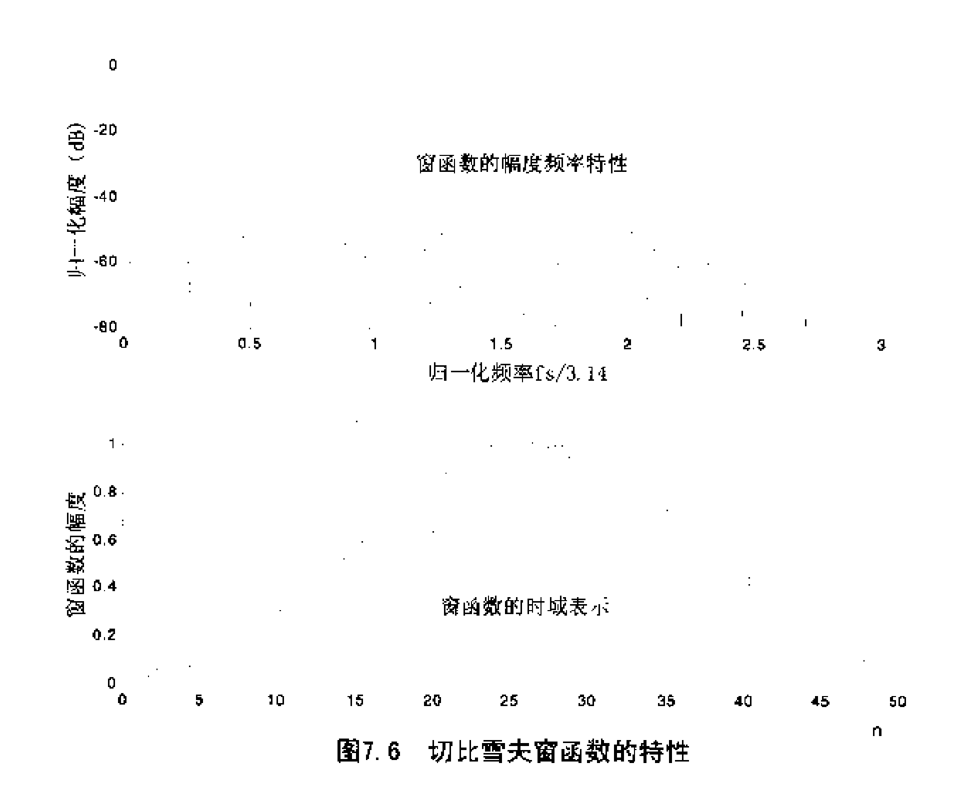


图7.6　切比雪夫窗函数的特性

7.1.7 巴特里特窗

巴特里特窗函数的时域形式可以表示为：

当 n 为奇数时：

$$w(k)=\begin{cases}\dfrac{2(k-1)}{n-1} & 1\leqslant k\leqslant \dfrac{n+1}{2}\\[2mm] 2-\dfrac{2(k-1)}{n-1} & \dfrac{n+1}{2}\leqslant k\leqslant n\end{cases}$$

当 n 为偶数时：

$$w(k)=\begin{cases}\dfrac{2(k-1)}{n-1} & 1\leqslant k\leqslant \dfrac{n}{2}\\[2mm] \dfrac{2(n-k)}{n-1} & \dfrac{n}{2}+1\leqslant k\leqslant n\end{cases}$$

它的幅度频率特性和时域特性如图 7.7 所示(选择窗函数的长度为 51)。

在 MATLAB 中可以利用 w = bartlett (n)的形式得到窗函数，其中 n 是窗函数的长度，而返回值 w 是一个 n 阶的向量，包含了窗函数的 n 个系数。

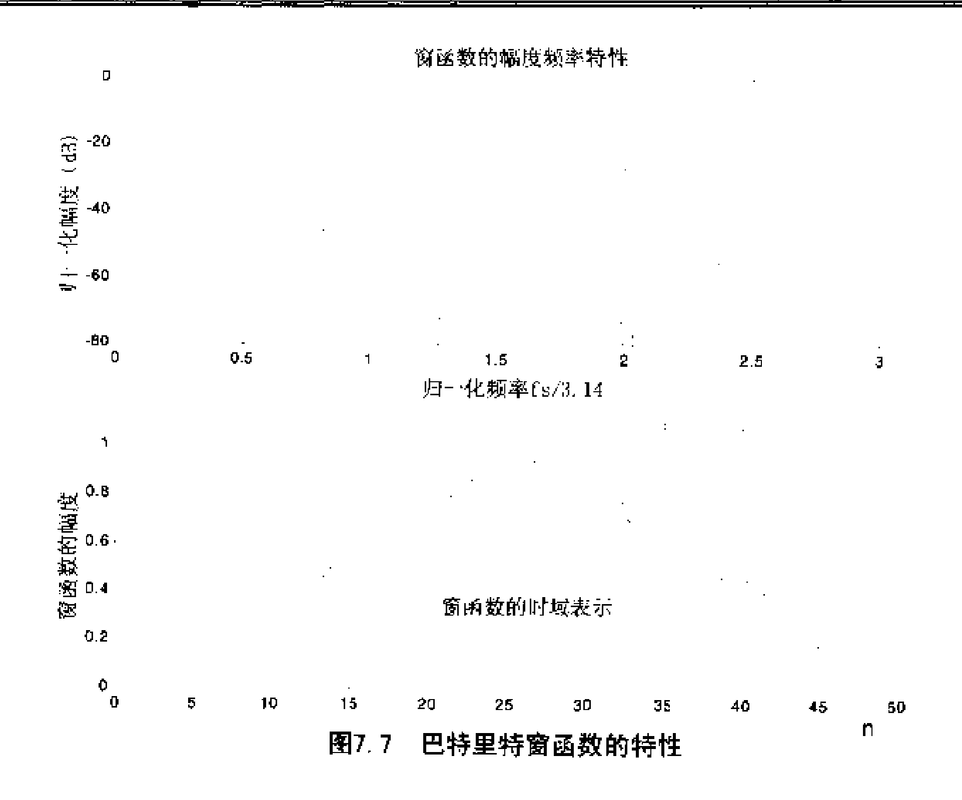


图7.7 巴特里特窗函数的特性

7.1.8 凯塞窗

还有一种适应能力比较强的窗函数叫做凯塞窗函数，其窗函数的时域形式可以表示为：

$$w(n)=\frac{I_0\left(\beta\sqrt{1-\left[1-\frac{2n}{N-1}\right]^2}\right)}{I_0(\beta)} \qquad 0\leqslant n\leqslant N-1$$

其中 $I_0(\beta)$是第1类变形零阶贝塞尔函数，β是窗函数的形状的参数，可以根据下式计算得到：

$$\beta=\begin{cases}0.1102(\alpha-8.7) & a>50\\ 0.5482(\alpha-21)^{0.4}+0.07886(\alpha-21) & 21\leqslant a\leqslant 50\\ 0 & a<21\end{cases}$$

其中的α为主瓣值和旁瓣值之间的差值(dB)。β值越大，窗函数的频谱的旁瓣值就越小，而主瓣的宽度就越宽。因而，改变β的取值，可以对主瓣宽度和旁瓣衰减进行选择。

在MATLAB中可以利用w = kaiser(n,beta)的形式得到窗函数，其中n是窗函数的长度，而返回值w是一个n阶的向量，包含了窗函数的n个系数，beta为窗函数的参数β。

图7.8显示了凯塞窗函数的幅度频率特性和它的时域特性，其中β取值为7.856。

在表7.1中对矩形窗、三角窗、布拉克曼窗、汉宁窗、海明窗、凯塞窗函数的性能进行了比较，在设计FIR滤波器的过程中可以根据要求选择合适的窗函数进行滤波器的设计。

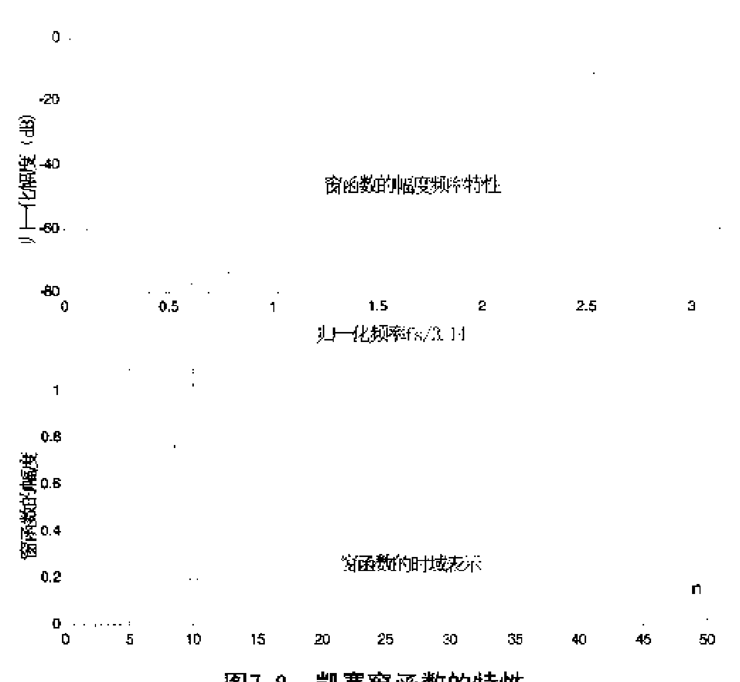


图7.8　凯塞窗函数的特性

表 7.1　各种窗函数的性能比较

窗函数	旁瓣峰值幅度(dB)	过渡带宽度	阻带最小衰减(dB)
矩形窗	-13	4π/N	−21
三角窗	-25	8π/N	−25
汉宁窗	-31	8π/N	−44
海明窗	-41	8π/N	−53
布拉克曼窗	-57	12π/N	−74
凯塞窗(β=7.856)	-57	10π/N	−80

7.2　用窗函数设计 FIR 数字滤波器

设计 FIR 数字滤波器的最简单的方法是窗函数法，通常也称之为傅立叶级数法。

FIR 数字滤波器的设计同 IIR 数字滤波器的设计一样，首先给出要求的理想滤波器的频率响应 $H_d(e^{j\omega})$，设计一个 FIR 数字滤波器频率响应 $H(e^{j\omega})$，去逼近理想的频率响应 $H_d(e^{j\omega})$。然而，窗函数法设计 FIR 数字滤波器是在时域进行的，因而必须由理想的频率响应 $H_d(e^{j\omega})$ 推导出对应的单位取样响应 $h_d(n)$，在设计一个 FIR 数字滤波器的单位取样响应 $h(n)$ 去逼近 $h_d(n)$。

以设计一个截止频率为 ω_c 的理想低通数字滤波器为例，要求滤波器具有线性相位，它的幅度频率特性如图 7.9a 所示。

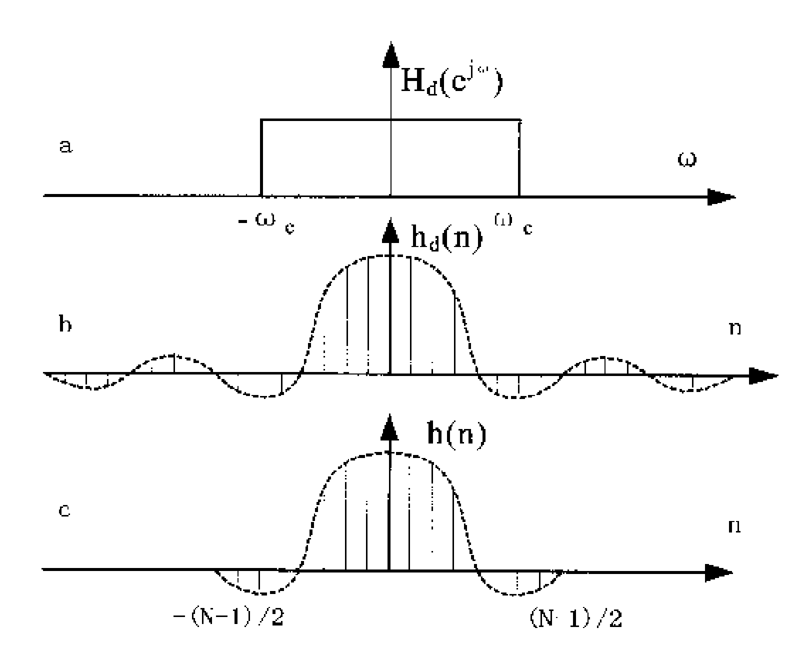


图7.9　FIR数字滤波器窗函数设计思想

由 $H_d(e^{j\omega})$的傅立叶反变换可以推导出 $h_d(n)$:

$$h_d(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d\left(e^{j\omega}\right)e^{jn\omega}d\omega$$

由于 $H_d(e^{j\omega})$的矩形特性，$h_d(n)$必然是无限长的，又是非因果的。而待设计的 FIR 数字滤波器，要求它的单位取样响应 h(n)是有限长的，而且是因果的。因此，要拥有有限长的 h(n)去逼近无限长的 $h_d(n)$，显然必须解决两个问题，一是有限长问题，最简单的办法是直接截短 $h_d(n)$，使得 $h_d(n)$和 h(n)之间的关系为：

$$h(n)=\begin{cases}h_d(n) & -\frac{(N-1)}{2}\leqslant n\leqslant\frac{(N-1)}{2}\\ 0 & \text{其余的 } n\end{cases}$$

式中的 N 为奇数。这个截短过程可以认为是无限长的取样响应和有限长的窗函数 w(n)的乘积，即：

$$h(n)=h_d(n)w(n)$$

但是这种简单的截短过程不能得到因果的 h(n)。

假设低通特性有群延迟 a，即：

$$H_d\left(e^{j\omega}\right)=\begin{cases}e^{-ja\omega} & |\omega|\leqslant\omega_c\\ 0 & \omega_c<|\omega|\leqslant\pi\end{cases}$$

在这里，若取 a=(N-1)/2，表明 $H_d(e^{j\omega})$的相位特性为 ωa，则可以得到 $h_d(n)$的表达式：

$$\begin{aligned}h_d(n)&=\frac{1}{2\pi}\int_{-\pi}^{\pi}e^{-ja\omega}e^{jn\omega}d\omega\\&=\frac{\sin[\omega_c(n-a)]}{\pi(n-a)}\end{aligned}$$

这是一个中心点在 a 的偶对称、无限长的、非因果序列，如图 7.10a 所示。

为了得到有限长序列，最简单的方法是选取矩形窗函数 $R_N(n)$，如图 7.10b 所示，即：

$$w(n)=R_N(n)=\begin{cases}1 & 0\leqslant n\leqslant N-1\\ 0 & 其他的\ n\end{cases}$$

为了保证所得到的滤波器是线性相位的滤波器，h(n)必须满足偶对称性，h(n)=h(N-1-n)。由于群延迟 a=(N-1)/2，只要截取从 n=0 到 n=N-1 的一段作为 h(n)。即 a 应该是 h(n)长度的一半。由此可以得到所需的滤波器的 h(n)：

$$\begin{aligned}h(n)&=h_d(n)w(n)\\&=\begin{cases}h_d(n) & 0\leqslant n<N-1\\ 0 & 其他的\ n\end{cases}\end{aligned}$$

如图 7.10c 所示。

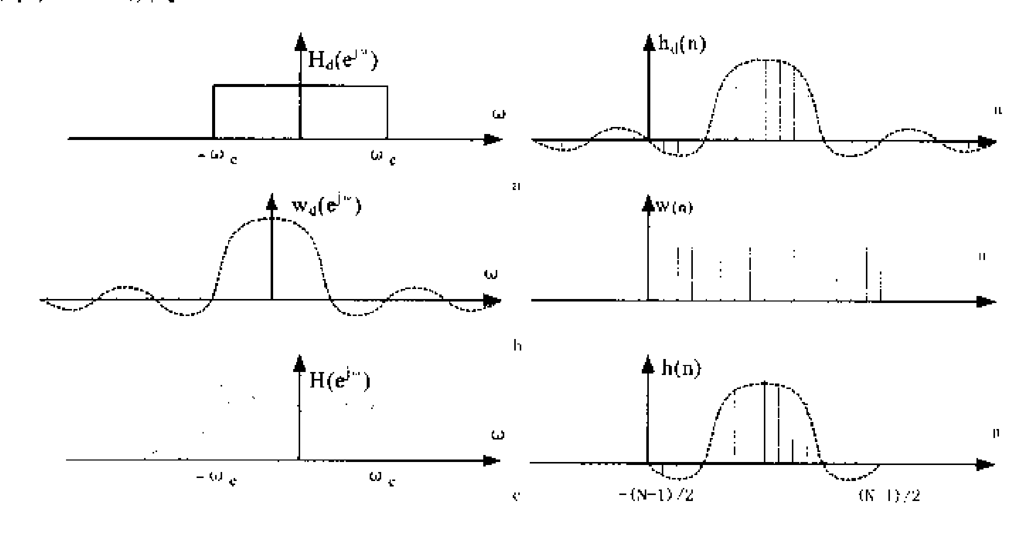


图7.10 矩形窗函数及因果h(n)的截取

由复卷积定理可知，时域相乘，频域是周期的卷积关系，所以 h(n)的频率特性为：

$$H\left(e^{j\omega}\right)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d\left(e^{j\theta}\right)W\left(e^{j(\omega-\theta)}\right)d\theta$$

$H(e^{j\omega})$是否能够很好地逼近 $H_d(e^{j\omega})$取决于窗函数的频谱特性 $W(e^{j\omega})$，在此处选用了矩形窗函数，它的频率特性为：

$$\begin{aligned}W_R\left(e^{j\omega}\right)&=\frac{1-e^{-jN\omega}}{1-e^{-j\omega}}\\&=e^{-j\omega\left(\frac{N-1}{2}\right)}\frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\end{aligned}$$

幅度特性和相位特性分别是：

$$W_R(\omega)=\left|W_R\left(e^{j\omega}\right)\right|=\frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}$$

和

$$Arg\left(W_R\left(e^{j\omega}\right)\right)=\omega\left(\frac{N-1}{2}\right)$$

由幅度特性可以看到，它的主瓣宽度为4π/N。若将理想滤波器的频率响应写成：

$$H_d\left(e^{j\omega}\right)=H_d(\omega)e^{-j\left(\frac{N-1}{2}\right)\omega}$$

其中幅度频率特性：

$$H_d(\omega)=\begin{cases}1 & |\omega|\leq\omega_c\\ 0 & \omega_c<|\omega|\leq\pi\end{cases}$$

得到的滤波器的频率特性是理想滤波器的频率特性和矩形窗函数的频率特性的卷积，即：

$$H\left(e^{j\omega}\right)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\theta)e^{-j\left(\frac{N-1}{2}\right)\theta}W_R\left((\omega-\theta\right)e^{-j(\omega-\theta)\left(\frac{N-1}{2}\right)}d\theta$$

$$=e^{-j\left(\frac{N-1}{2}\right)\omega}\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\theta)W_R\left((\omega-\theta\right)d\theta$$

由此可以得到所设计的滤波器的幅度频率特性为：

$$H(\omega)=\left|H\left(e^{j\omega}\right)\right|=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(\theta)W_R\left((\omega-\theta\right)d\theta$$

由上式可见，对实际FIR滤波器H(ω)有影响的只是窗函数的幅度频率特性 $W_R(\omega)$。实际中的FIR滤波器的幅度频率特性是理想低通滤波器的幅度频率特性和窗函数的幅度频率特性的复卷积。复卷积给H(ω)带来过冲和波动，所以加窗函数后，对滤波器的理想特性的影响有以下几点：

- $H_d(\omega)$在截止频率的间断点变成了连续的曲线，使得H(ω)出现了一个过渡带，它的宽度等于窗函数的主瓣的宽度。由此可知，如果窗函数的主瓣越宽，过渡带就越宽。
- 由于窗函数旁瓣的影响，使得滤波器的幅度频率特性出现了波动，波动的幅度取决于旁瓣的相对幅度。旁瓣范围的面积越大，通带波动和阻带的波动就越大，换句话说，阻带的衰减减小。而波动的多少，取决于旁瓣的多少。
- 增加窗函数的长度，只能减少窗函数的幅度频率特性 W(ω)的主瓣宽度，而不能减少主瓣和旁瓣的相对值，该值取决于窗函数的形状。换句话说，增加截取函数的长度N只能相应的减小过渡带，而不能改变滤波器的波动程度。

为了满足工程上的需要，可以通过改变窗函数的形状来改善滤波器的幅度频率特性，而窗函数的选择原则是：

- 具有较低的旁瓣幅度，尤其是第一旁瓣的幅度。
- 旁瓣的幅度下降的速率要快，以利于增加阻带的衰减。
- 主瓣的宽度要窄，这样可以得到比较窄的过渡带。

通常上述的几点难以同时满足。当选用主瓣宽度较窄时，虽然能够得到比较陡峭的幅度频率响应，但是通带和阻带的波动明显增加；当选用比较小的旁瓣幅度时，虽然能够得到比较平坦和匀滑的幅度频率响应，但是过渡带将加宽。因此，实际中选用的窗函数往往是它们的折衷。在保证主瓣的宽度达到一定要求的条件下，适当地牺牲主瓣的宽度来换取旁瓣的波动减小。以上是从幅度频率特性设计方面对窗函数提出的要求，实际中设计FIR数字滤波器往往要求是线性相位的，因此要求w(n)满足线性相位的条件，即要求w(n)满足

$$w(n)=w(N-1-n)$$

综上所述，窗函数不仅有截短的作用，而且能够起到平滑的作用，在很多领域得到了

应用。

下面通过利用第 1 节介绍的 8 种窗函数分别设计低通数字滤波器并分析它们的效果。信号的采样频率是 1000Hz，数字滤波器的截止频率是 200Hz，滤波器的阶数为 81。可以通过编写一个 MATLAB 程序得到结果。

程序清单如下：

```
passrad=0.4*pi;
w1=boxcar(81);
w2=triang(81);
w3=hamming(81);
w4=hanning(81);
w5=bartlett(81);
w6=blackman(81);
w7=chebwin(81,30);
w8=kaiser(81,7.856);
n=1:1:81;
hd=sin(passrad*(n-41))./(pi*(n-41));
hd(41)=passrad/pi;
h1=hd.*rot90(w1);
h2=hd.*rot90(w2);
h3=hd.*rot90(w3);
h4=hd.*rot90(w4);
h5=hd.*rot90(w5);
h6=hd.*rot90(w6);
h7=hd.*rot90(w7);
h8=hd.*rot90(w8);
[mag1,rad]=freqz(h1);
[mag2,rad]=freqz(h2);
[mag3,rad]=freqz(h3);
[mag4,rad]=freqz(h4);
[mag5,rad]=freqz(h5);
[mag6,rad]=freqz(h6);
[mag7,rad]=freqz(h7);
[mag8,rad]=freqz(h8);
subplot(2,2,1);
plot(rad,20*log10(abs(mag1)));
grid on;
subplot(2,2,2);
plot(rad,20*log10(abs(mag2)));
grid on;
subplot(2,2,3);
plot(rad,20*log10(abs(mag3)));
grid on;
subplot(2,2,4);
plot(rad,20*log10(abs(mag4)));
grid on;
figure;
subplot(2,2,1);
plot(rad,20*log10(abs(mag5)));
grid on;
```

```
subplot(2,2,2);
plot(rad,20*log10(abs(mag6)));
grid on;
subplot(2,2,3);
plot(rad,20*log10(abs(mag7)));
grid on;
subplot(2,2,4);
plot(rad,20*log10(abs(mag8)));
grid on;
```

可以得到利用各种窗函数设计得到的低通滤波器的幅度频率特性，如图 7.11 所示。

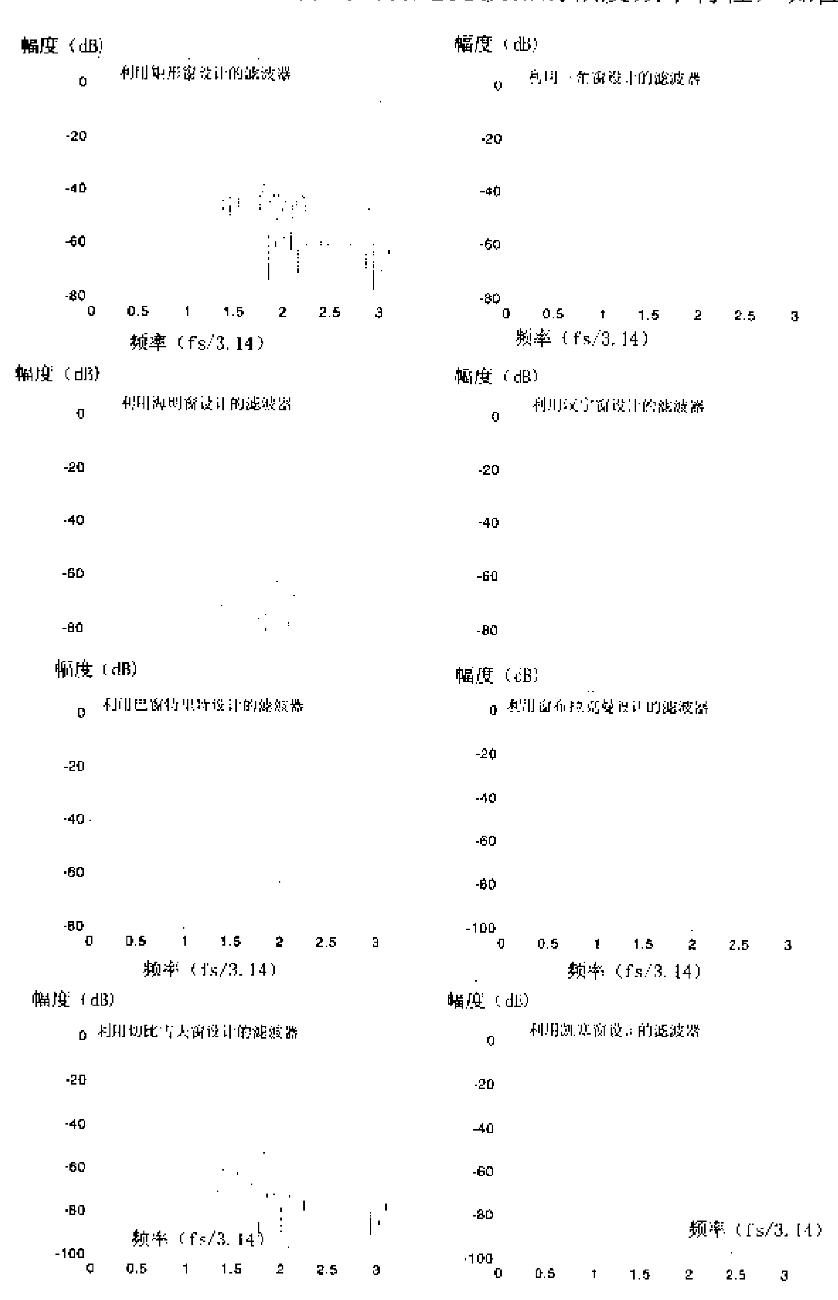


图7.11　利用各种窗函数设计的低通滤波器

7.3 用频率抽样法设计 FIR 滤波器

窗函数设计 FIR 数字滤波器是从时域出发，把理想的滤波器的单位取样响应 $h_d(n)$用合适的窗函数截短成为有限长度的 h(n)，使得 h(n)逼近理想的 $h_d(n)$。以实现滤波器的频率响应 $H(e^{j\omega})$逼近要求的理想滤波器的频率响应 $H_d(e^{j\omega})$。

一个有限长的序列，如果满足频域采样定理的条件，可以通过频谱的有限个采样点的值准确地恢复。

所谓用频率取样法设计 FIR 数字滤波器是从频域出发，根据频域的采样定理，对给定的理想滤波器的频率响应 $H_d(e^{j\omega})$进行等间隔的采样：

$$H_d\left(e^{j\omega}\right)\Big|_{\omega=(2\pi)k/N}=H_d(k) \qquad k=0,1,\cdots N-1$$

把 $H_d(k)$当作待设计的 FIR 滤波器的频率相应的采样值 H(k)，即：

$$H(k)=H_d(k)=H_d\left(e^{j\omega}\right)\Big|_{\omega=(2\pi)k/N} \qquad k=0,1,\cdots N-1$$

利用频域的 N 个采样点的值 H(k)，通过下式可以求出滤波器的系统函数 H(z)和频率响应 $H(e^{j\omega})$：

$$H(z)=\frac{1-z^{-N}}{N}\sum_{k=0}^{N-1}\frac{H(k)}{1-W_N^{-k}z^{-1}}$$

$$H(e^{j\omega})=\sum_{k=0}^{N-1}H(k)\phi\left(\omega-\frac{2\pi}{N}k\right)$$

其中 φ(w)是一个内插函数：

$$\phi(w)=\frac{1}{N}\frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}e^{-j\omega(N-1)/2}$$

$$W_N=e^{j\frac{2\pi}{N}}$$

考虑到 $e^{jk\pi}=e^{-jk\pi}=(-1)^k$，对 $H(e^{j\omega})$的表达是进行化简得到：

$$H(e^{j\omega})=e^{-j\left(\frac{N-1}{2}\right)\omega}\sum_{k=0}^{N-1}H(k)\frac{(-1)^k}{N}e^{-j\frac{\pi k}{N}}\frac{\sin\left(\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)N\right)}{\sin\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)}$$

$$=e^{-j\left(\frac{N-1}{2}\right)\omega}\sum_{k=0}^{N-1}H(k)\phi_k(\omega)$$

在不同的采样点上的内插函数 $\phi_k(w)$为：

$$\phi_k(\omega)=e^{-j\frac{\pi k}{N}}\frac{\sin\left(\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)N\right)}{\sin\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)}$$

从上面的两个公式可以看出，在各个采样频率点 $\omega_k=2\pi k/N$ 处，待设计的 FIR 滤波器

的频率响应严格等于采样值 H(k)。

$$H\left(e^{jk\frac{2\pi}{N}}\right)=H(k)=H_d(k)=H_d\left(e^{jk\frac{2\pi}{N}}\right)$$

而在各采样点间的频率响应则是各个采样点的内插函数延伸叠加的结果。

对于一个无限长的序列，根据它的频谱的有限个采样值是不能准确地将其频谱恢复的，换句话说，由于频谱的有限个采样值恢复出来的频率响应实际上是一个对理想频率相应的逼近。因此这种方法必然有一定的逼近误差。若被逼近的频率响应比较平滑，则各采样点之间的逼近误差较小。反之，则逼近的误差比较大。图 7.12 显示了这个结果。图中的虚线表示理想滤波器的频率响应 $H_d(e^{j\omega})$，圆点表示它的采样值 H(k)，实线表示 H(k)的连续内插，也就是 $H(e^{j\omega})$，从图 7.12a 中理想滤波器的频率响应是一个矩形，在通带和阻带之间不是连续的，变化剧烈，所以设计得到的滤波器逼近的效果较差。而图 7.12b 中，理想滤波器的频率响应在通带和阻带之间有过渡带，所以设计得到的滤波器的频率响应的逼近的效果较好。

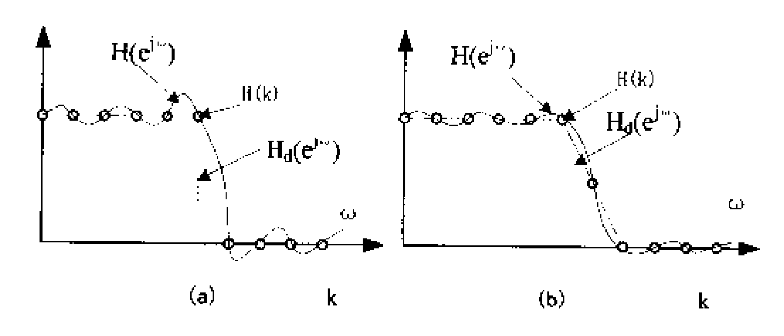


图7.12 频率采样法及其连续内插

为了提高逼近的质量，减少逼近误差，可以采用人为地扩展过渡带的方法，即在频率相应的过渡带内插入一个或多个比较连续的采样点，使得过渡带比较连续，从而使得通带和阻带之间变法比较缓慢，使得设计得到的滤波器对理想滤波器的逼近误差较小。

因为设计得到的滤波器的频率响应是各个频域采样值 H(k)的连续内插的叠加结果，而内插函数与 $\dfrac{\sin\left(\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)N\right)}{\sin\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)}$ 成正比，所以在过渡带内增补的采样值的内插函数(即 $\dfrac{\sin\left(\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)N\right)}{\sin\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)}$)对相邻的通带和阻带的波动产生良好的抵消作用。只要进行设计过渡带的采样值，就可以改善通带和阻带的波动，从而设计出性能良好的滤波器。当然，设计出来的滤波器的过渡带也会加宽。

过渡带内采样值的选择可以利用计算机按照梯度法进行搜索，根据求解通带和阻带内的一组约束方程进行。这组约束方程式为：

通带 $\left|H\left(e^{j\omega}\right)-H_d\left(e^{j\omega}\right)\right|\leq\sigma$

阻带 $\max\left|H\left(e^{j\omega}\right)-H_d\left(e^{j\omega}\right)\right|=\min(H_1或H_1,H_2)$

式中的 σ 是固定的容差。频率取样法可以做到最大误差的最小近似。

在理想低通滤波器的设计中，若不增加过渡点，阻带和通带之间的衰减约为−21dB，如果在通带和阻带之间增加一个采样点，阻带的最小衰减可以提高到−65dB，如果增加两个采样点，阻带的最小衰减可以提高到−75dB，如果增加3个采样点，阻带的最小衰减可以提高到−85dB至−95dB。

7.4 FIR滤波器的切比雪夫逼近法

窗函数法和频率取样法设计FIR数字滤波器都是比较有效的，同时它们都有固有的缺点。窗函数法不容易设计预定给出截止频率的滤波器，也不能够解出在给定滤波器的阶数N时，怎样才能够设计出一个最佳的FIR数字滤波器的问题。频率取样法是一种优化设计方法，但是赖以进行优化设计的变量只限于过渡带上的几个采样值，因而它不是最优设计。

从FIR 数字滤波器的系统函数可以看出，极点都是在z平面的原点，而零点的分布是任意的。不同的分布将对应不同的频率响应，最优设计实际上就是调节这些零点的分布，使得实际滤波器的频率响应$H(e^{j\omega})$和理想滤波器的频率响应$H_d(e^{j\omega})$之间的最大绝对误差最小。在本节将讨论一种频域设计的最佳方法——等波纹切比雪夫法。它是采用最大误差最小准则得到最佳数字滤波器，并且最佳解是唯一的。

通常要求线性相位滤波器在不同的频带内逼近的容差不同。例如，通带能最大容许误差要求不超过$\pm\sigma_1$，阻带内最大容许误差不超过$\pm\sigma_2$。切比雪夫等波纹逼近是采用加权逼近误差$E(e^{j\omega})$，它可以表示为：

$$E\left(e^{j\omega}\right)=W\left(e^{j\omega}\right)\left(H\left(e^{j\omega}\right)-H_d\left(e^{j\omega}\right)\right)$$

其中$W(e^{j\omega})$是一个加权函数，在误差要求高的频段上，可以取比较大的加权值，否则，取较小的加权值。理想频率响应和加权函数分别表示为：

$$\left|H_d(e^{j\omega})\right|=\begin{cases}1 & |\omega|\leqslant\omega_c\\ 0 & \omega_c<|\omega|\leqslant\pi\end{cases}$$

$$W(e^{j\omega})=\begin{cases}1/k & |\omega|\leqslant\omega_c\\ 1 & \omega_c<|\omega|\leqslant\pi\end{cases}$$

上式中的常数$k=\sigma_1/\sigma_2$。

为了保证设计出的$H(e^{j\omega})$具有线性相位，这里仍然要遵循线性相位的约束条件，即：

$$h(n)=\pm h(N-1-n)\qquad 0\leqslant n\leqslant N-1$$

按照FIR数字滤波器的单位脉冲响应h(n)的对称性和N的奇、偶性，FIR数字滤波器可以分为4种类型，如表7.2所示。

滤波器的频率响应可以写成统一的形式：

$$H\left(e^{j\omega}\right)=e^{-j\left(\frac{N-1}{2}\right)\omega}e^{j\frac{\pi}{2}k}\hat{H}\left(e^{j\omega}\right)$$

上式中的K和$\hat{H}\left(e^{j\omega}\right)$的值可以在表7.2中找到。从$\hat{H}\left(e^{j\omega}\right)$的表达式可以看出，它是一个纯实数。

表 7.2　4 种类型的线性相位函数

参数 / 类型	K	$\hat{H}(e^{j\omega})$
1—N 为奇数，偶对称脉冲响应	0	$\sum_{n=0}^{(N-1)/2} a(n)\cos(n\omega)$
1—N 为偶数，奇对称脉冲响应	0	$\sum_{n=0}^{(N)/2} b(n)\cos((n-1/2)\omega)$
1—N 为奇数，偶对称脉冲响应	1	$\sum_{n=0}^{(N-1)/2} c(n)\sin(n\omega)$
1—N 为偶数，奇对称脉冲响应	1	$\sum_{n=0}^{(N)/2} d(n)\sin((n-1/2)\omega)$

在表 7.3 中，$\hat{H}(e^{j\omega})$的每个表达式都可以写成统一的形式：

$$\hat{H}(e^{j\omega})=Q(e^{j\omega})P(e^{j\omega})$$

其中 $Q(e^{j\omega})$为 ω 的固定函数，而 $P(e^{j\omega})$为 M 个余弦函数的线性组合，4 类线性相位 FIR 的 $Q(e^{j\omega})$和 $P(e^{j\omega})$的表达式见表 7.3。线性相位滤波器 4 种类型中的求和是统一为余弦的线性组合，从而有利于采用同一种算法进行最佳逼近。

表 7.3　4 种线性相位的 $\hat{H}(e^{j\omega})$

	$Q(e^{j\omega})$	$P(e^{j\omega})$
类型 1	1	$\sum_{n=0}^{(N-1)/2} a(n)\cos(n\omega)$
类型 2	$\cos\left(\frac{\omega}{2}\right)$	$\sum_{n=0}^{(N-2)/2} \tilde{b}(n)\cos(n\omega)$
类型 3	$\sin(\omega)$	$\sum_{n=0}^{(N-3)/2} \tilde{c}(n)\cos(n\omega)$
类型 4	$\sin\left(\frac{\omega}{2}\right)$	$\sum_{n=0}^{(N)/2} \tilde{d}(n)\cos(n\omega)$

切比雪夫等波纹逼近采用的加权逼近误差 $E(e^{j\omega})$可以改写成：

$$E(e^{j\omega})=W(e^{j\omega})\,Q(e^{j\omega})\left(P(e^{j\omega})-\frac{H_d(e^{j\omega})}{Q(e^{j\omega})}\right)$$

若令：

$$\hat{W}(e^{j\omega})=W(e^{j\omega})Q(e^{j\omega})$$

$$\hat{H}_d(e^{j\omega})=\frac{H_d(e^{j\omega})}{Q(e^{j\omega})}$$

可以得到误差函数的表达式为：

$$E(e^{j\omega})=\hat{W}(e^{j\omega})(P(e^{j\omega})-\hat{H}_d(e^{j\omega}))$$

因此，最优滤波器设计就是在最大误差最小化的准则下，求出滤波器系数组($\tilde{a}(n)$、$\tilde{b}(n)$、$\tilde{c}(n)$和$\tilde{d}(n)$)。切比雪夫逼近问题可以这样描述：在实行逼近的频率范围内(即滤波器的通带和阻带)，使得误差函数$E(e^{j\omega})$的最大绝对值为最小来确定线性相位滤波器的系数组($\tilde{a}(n)$、$\tilde{b}(n)$、$\tilde{c}(n)$和$\tilde{d}(n)$)。

若用符号$\|E(e^{j\omega})\|$表示这个最小值，则切比雪夫逼近公式可以写成：

$$\|E(e^{j\omega})\|=\min[\max(|E(e^{j\omega})|)]$$

换句话说，对于切比雪夫逼近问题只要求解上式就可以了，便可以得到最优设计。其中可以利用切比雪夫逼近问题的一个重要性质——交替定理进行求解。

近几年来，人们为了寻求最优化设计作了大量的工作。尤其 1973 年出现了用雷米兹(REMEZ)算法求解加权误差值$\|E(e^{j\omega})\|=\min[\max(|E(e^{j\omega})|)]$的方程，使得滤波器的阶数、通带和阻带的边缘误差以及误差的加权函数都可以自由选择。较好的满足了设计需要，成为当前设计线性相位 FIR 数字滤波器的一种最好的 CAD 设计方法。

雷米兹算法的思路是：

首先在滤波器的通带和阻带上等间隔地取 M+1 个频率点，ω_0、ω_1、ω_2……ω_M，作为极值点的初始猜测点，然后求解满足下式的σ值：

$$\hat{W}(e^{j\omega})(P(e^{j\omega})-\hat{H}_d(e^{j\omega}))=(-1)^k\sigma$$

但是并不知道$P(e^{j\omega})$的值，可以假设：

$$P(e^{j\omega})=\sum_{n=0}^{M}a(n)\cos(n\omega)$$

代入上式便可解出σ的值。

接着利用求出来的σ值和预先的 M+1 各频率点，求出$P(e^{j\omega_k})$，其中 k=0，1，2，3……M，然后利用拉格郎日插值公式，求出$P(e^{j\omega})$的表达式。

接着利用求得的$P(e^{j\omega})$，计算误差：

$$E(e^{j\omega})=\hat{W}(e^{j\omega})(P(e^{j\omega})-\hat{H}_d(e^{j\omega}))$$

并检验是否能够满足$|E(e^{j\omega})|\le\sigma$，若满足这个要求，则说明已经得到最优的解。若在某些频率上$|E(e^{j\omega})|>\sigma$，则把这些频率点作为新的极值点，$\omega 0$、$\omega 1$、$\omega 2$……ωM，重新进行计算。经过反复的迭代，直到能够满足$|E(e^{j\omega})|\le\sigma$为止，这时的$\sigma$就是要求的$\sigma$的最小值，因而达到了最优逼近。

上述算法的结果得到 $E(e^{j\omega})$，还要经过傅立叶反变换求得单位脉冲响应，因而，切比雪夫逼近法设计 FIR 数字滤波器的程序是：

(1) 规定所需的频率响应$H_d(e^{j\omega})$，加权函数$W(e^{j\omega})$和滤波器的单位脉冲响应的长度 N。

(2) 形成$\hat{P}(e^{j\omega})$、$\hat{W}(e^{j\omega})$和$P(e^{j\omega})$。

(3) 利用雷米兹多重交换算法求解逼近问题。

(4) 计算滤波器的单位脉冲响应 h(n)。

利用非线性规划中的雷米兹算法进行 FIR 滤波器的优化设计有很多优点，但是设计指标上不仅有频率上的要求，有时又有时间响应上的约束，这时雷米兹算法就不适用了，这时线性规划法就是唯一有效的方法了。

采用切比雪夫逼近设计方法能够得到既有严格线性相位，又有很好衰减特性的滤波器，因此，切比雪夫逼近法在滤波器设计中占有很重要的位置。

在 MATLAB 的 Toolbox 中有一个 remez 函数，采用了雷米兹算法，可以直接利用这个函数进行滤波器的设计。在接下来的 7.5 节将对这个函数进行介绍。

7.5　利用 MATLAB 设计 FIR 滤波器

在 MATLAB 中一共有 5 种设计 FIR 数字滤波器的方法。第 1 种是窗函数法，它的设计思路和前面介绍的窗函数设计法相同，对应的 MATLAB 函数有 fir1、fir2、kaiserord；第 2 种方法是含过渡带的设计方法，它利用等波纹或者最小均方取逼近理想滤波器的频域响应，其中包含了切比雪夫逼近设计方法，对应的 MATLAB 函数有 firls、remez、remezord；第 3 种方法是最小二乘约束设计方法，它使得设计的滤波器和理想滤波器的误差在整个频段上积分，使得积分值最小，对应的 MATLAB 函数有 fircls、fircls1；第 4 种方法是非线性相位滤波器设计方法，它设计出的 FIR 滤波器是非线性相位的，对应的 MATLAB 函数为 cremez；第 5 种方法是升余弦方法，采用升余弦函数将进行滤波器设计，对应的 MATLAB 函数为 firrcos。根据前面章节的理论讨论结果，在本节中将讨论如何利用 fir1、fir2、kaiserord、remez 等函数进行滤波器的设计。

7.5.1　利用 fir1 函数设计 FIR 数字滤波器

在 MATLAB 中可以利用 fir1 函数设计各类 FIR 数字滤波器，包括低通、带通、高通、带阻等类型的滤波器。fir1 函数设计滤波器实际上是采用了窗函数设计 FIR 线性相位滤波器的方法，它的具体算法是：如果 w(n)为指定的窗，其中 1≤n≤N，理想滤波器的单位脉冲响应为 h(n)，预示可以得到设计的滤波器的系数为：b(n)=w(n)h(n)，1≤n≤N。对于采用窗函数设计任意频率响应的 FIR 滤波器则可以利用 fir2 函数，这个函数将在 7.5.3 小节进行介绍。

在 MATLAB 中，利用 fir1 函数设计各类 FIR 数字滤波器的具体语法形式如下：

```
b = fir1(n,Wn)
b = fir1(n,Wn,'ftype')
b = fir1(n,Wn,window)
b = fir1(n,Wn,'ftype',window)
b = fir1(...,'noscale')
```

对于 b = fir1(n,Wn)形式，输入的参数是滤波器的阶数 n 和滤波器的截止频率 Wn，截止频率 Wn 是一个 0 到 1 的数，1 对应为信号采样频率的一半。在这种情况下，采用默认的窗函数为海明窗函数进行滤波器的设计。函数的返回值 b 是一个 n+1 的向量，它包含了设计得到的滤波器的 n+1 个系数。得到的 FIR 数字滤波器是截止频率为 Wn 的线性相位低通滤波器，它的系统函数可以表示为：

$$B(z)=b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}$$

如果 Wn 是一个含有两个数的向量，Wn = [w1 w2]，fir1 函数则返回一个 n 阶带通滤波

器的系数，带通滤波器的通带范围为：w1<W<w2。

如果 Wn 是一个含有多个数的向量，Wn = [w1 w2 w3 w4 w5 ... wn]，fir1 函数则返回一个 n 阶的多带线性相位 FIR 滤波器的系数，这个滤波器的通带范围为：$0 < \omega < w1$、$w2 < \omega < w3$、...，$wn < \omega < 1$。

对于 b=fir1(n,Wn,'ftype')形式，可以通过改变参数 ftype 的值设计高通滤波器或者带阻滤波器。当 ftype='high'时，设计得到的滤波器是截止频率为 Wn 的高通滤波器；当 Wn = [w1 w2]，ftype = 'stop'时，设计得到的滤波器是阻带范围为 $w1<\omega<w2$ 的带阻滤波器。如果 Wn 是一个含有多个数的向量，Wn = [w1 w2 w3 w4 w5 ... wn]，得到的滤波器将是多带滤波器，这时如果设置 ftype = 'DC－1'默认值，滤波器的通带范围为：$0 < \omega < w1$、$w2 < \omega < w3$、...，$wn < \omega < 1$；这时如果设置 ftype = 'DC－0'，滤波器的通带范围为：$w1 < \omega < w2$、$w3 < \omega < w4$、...，$w_{(n-1)} < \omega < w_n$。

值得注意的是，fir1 在设计带阻和高通滤波器得到的滤波器的阶数总是偶数阶的，这是因为如果滤波器的阶数为奇数，它的频率响应在 $\omega = \pi$ (对应于采样频率的二分之一处)的值总是 0，显然不能满足高通和带阻滤波器的要求。如果设定的滤波器的阶数 n 这个参数为奇数，fir1 函数将自动增加滤波器的阶数为 n+1。

另外，可以指定滤波器设计中采用的窗函数的形式，具体的语法形式是：b = fir1(n,Wn, window)，参数 window 必须是一个长度为 n+1 的向量，如果没有指定窗函数，fir1 采用海明窗为默认的窗函数。

例，设计一个阶数为 48，通带范围为 0.35≤w≤0.65 的带通 FIR 线性相位滤波器，并分析它的频率特性，可以利用以下两条语句实现这个滤波器。

```
b = fir1(48,[0.35 0.65]);
freqz(b);
```

得到滤波器的频率特性如图 7.13 所示。

例，在 chirp.mat 文件中储存有信号 y 的数据，信号 y 的频率特性使它的大部分信号能量集中在 Fs/4 以上。设计一个 34 阶的 FIR 高通滤波器，滤除频率低于 Fs/4 的成分，其中采用的截止频率为 0.48(归一化后的频率)、阻带衰减为 30 Db 的切比雪夫窗函数；并且利用 sound 函数播放滤波前和滤波后的声音进行比较。程序的清单如下：

```
load chirp      % loads y and Fs
window= chebwin(35,30);
b = fir1(34,0.48,'high',window);
yfilt = filter(b,1,y);
[Py,fy] = pburg(y,10,512,Fs);
[Pyfilt,fyfilt] = pburg(yfilt,10,512,Fs);
plot(fy,10*log10(Py),':',fyfilt,10*log10(Pyfilt));
grid on
ylabel('Magnitude (dB)');
xlabel('Frequency (Hz)');
legend('Chirp','Filtered Chirp')
           sound(y,Fs);
sound(yfilt,Fs);
```

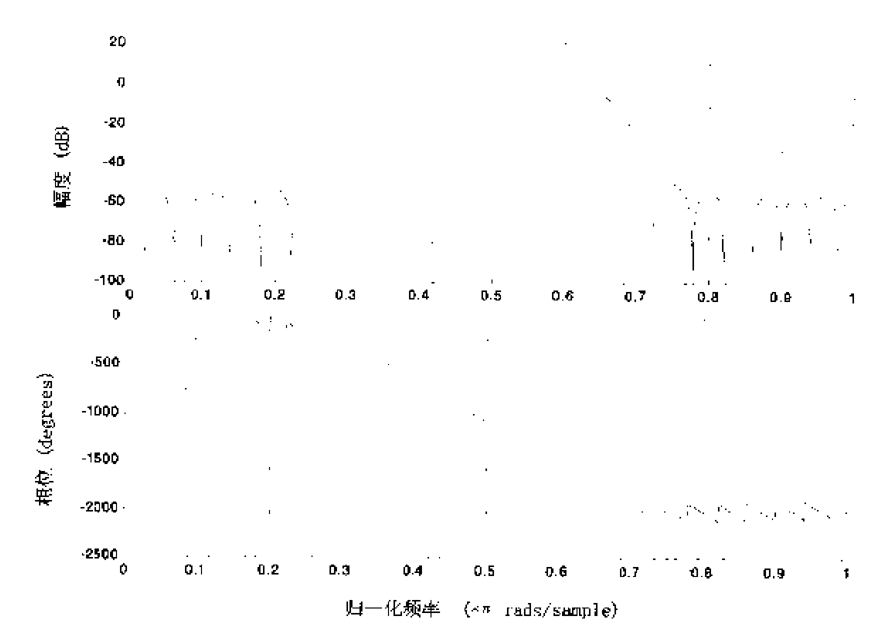


图7.13　利用fir1设计的带通滤波器的频率响应

得到滤波前的信号和滤波后的信号的比较如图 7.14 所示。

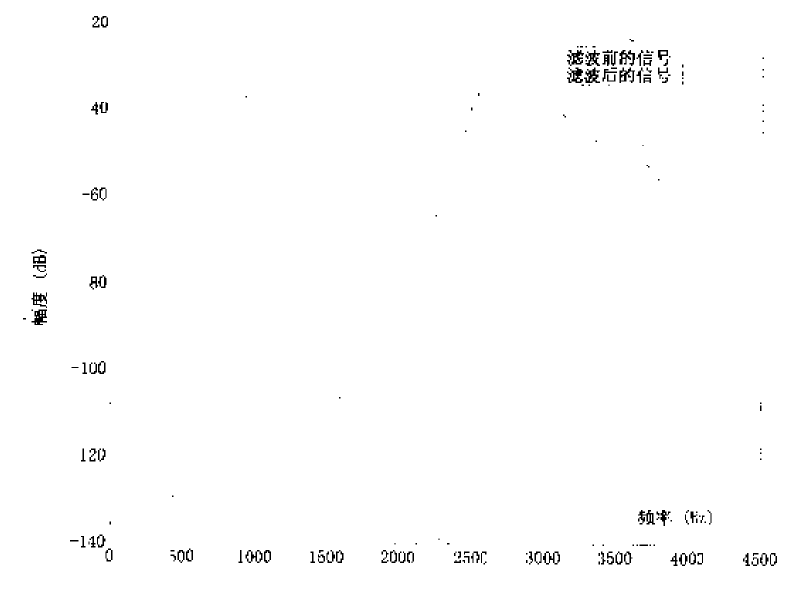


图7.14　信号滤波前后的频谱比较

7.5.2　利用 kaiserord 函数求取凯塞窗函数的参数

凯塞窗函数是一个性能比较良好，并且可以通过参数的控制来改变窗函数的主瓣带宽和旁瓣衰减，因此在用窗函数法设计 FIR 数字滤波器时，经常利用凯塞窗函数进行设计，但是凯塞窗函数参数必须预先设计。在 MATLAB 中可以利用 kaiserord 函数根据设计滤波器的要求得到这些所需的参数，然后进行凯塞窗函数的设计以及 FIR 滤波器的设计。Kaiserord 函数的具体语法形式如下：

```
[n,Wn,beta,ftype] = kaiserord(f,a,dev)
```

```
[n,Wn,beta,ftype] = kaiserord(f,a,dev,Fs)
c = kaiserord(f,a,dev,Fs,'cell')
```

下面具体介绍 kaiserord 函数的输入参数和返回值：

- 输入参数 f 是一个向量，其中的元素为待设计滤波器的过渡带的起始点和结束点，如果 kaiserord 中没有 Fs 参数，f 中的元素的取值范围从 0 到 1。例如，要求设计的滤波器的过渡带为 1000Hz 到 1200Hz、2800Hz 到 3000Hz，信号的采样频率为 8000Hz，则 f=[0.25 0.30 0.70 0.75]，如果 kaiserord 函数中含有第 4 个参数 Fs，则设定 Fs=8000，f=[1000 1200 2800 3000]。
- 输入参数 a 是一个向量，由参数 f 可以确定待设计滤波器除了过渡带的范围，也就是通带或者阻带，向量 a 中的元素指定这些频率段的理想的幅度值。例如，条件和前面的举例相同，若要求滤波器在 0Hz 到 1000Hz 范围内是通带，则这个频率段的理想幅度值为 1；在 1200Hz 到 2800Hz 是阻带，则在这个频率段的理想幅度值为 0；在 3000Hz 到 4000Hz 上是通带，则这个频率段的理想值幅度为 1，由此可以由这些值构成向量 a=[1 0 1]。可以看出，由参数 f 和 a 可以确定待设计滤波器的具体类型。
- 输入参数 dev 是一个向量，它的长度和 a 相同，其中的元素为各个通带和阻带内容许的幅度最大误差。条件和前面的举例相同，通带的容许误差为 0.01，阻带的容许误差为 0.02，则 dev=[0.01 0.02 0.01]。
- 返回值 n 是 kaiserord 函数根据待设计滤波器的要求，得到的能够满足设计要求的滤波器的最小阶数。可以根据下式求得：

$$n = \frac{\alpha - 7.95}{2.285(\Delta\omega)}$$

其中 $\alpha = -20\log_{10}\delta$ 是阻带的衰减的分贝表示形式。$\Delta\omega$ 是最小的过渡带范围。

- 返回值 Wn 是一个向量，它是 kaiserord 根据设计的要求，得到的滤波器的截止频率点。
- 返回值 beta 是 kaiserord 根据待设计的滤波器的要求，求出的 β 的数值。它是根据下式求得：

$$\beta = \begin{cases} 0.1102(\alpha - 8.7) & \alpha > 50 \\ 0.5482(\alpha - 21)^{0.4} + 0.07886(\alpha - 21) & 50 \geqslant \alpha \geqslant 21 \\ 0 & \alpha < 21 \end{cases}$$

- 返回值 ftype 是 kaiserord 根据待设计滤波器的要求得到的滤波器的类型，当 ftype ='high'时，滤波器是截止频率为 Wn 的高通滤波器；ftype ='stop'时，滤波器是带阻滤波器。如果设置 ftype = 'DC-1'，滤波器的通带范围为：$0 < \omega < w1$、$w2 < \omega < w3$、...，$wn < \omega < 1$；如果 ftype = 'DC-0'，滤波器的通带范围为：$w1 < \omega < w2$、$w3 < \omega < w4$、...，$w_{(n-1)} < \omega < w_n$。

利用 kaiserord 函数得到各种参数后，可以利用返回值 n 和 beta 设计凯塞窗函数，然后利用返回值 Wn 和 ftype 传输给 fir1 进行滤波器的设计。

例，利用凯塞窗函数设计一个低通 FIR 数字滤波器，通带的范围是 0Hz 到 1000Hz，阻带的范围是 1500Hz 到 4000Hz，通带的波纹最大为 0.05，阻带的波纹最大为 0.01。信号的

采样频率为 8000Hz。

程序的清单如下：

```
fsamp = 8000;
fcuts = [1000 1500];
mags = [1 0];
devs = [0.05 0.01];
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,fsamp);
hh = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
freqz(hh)
```

得到滤波器的阶数为 36，滤波器的截止频率为 0.3125，凯塞窗函数的 β 值为 3.3953。滤波器的系数为：

```
[-0.0024   -0.0031    0        0.0055   0.0078    0.0021   -0.0093   -0.016
 -0.0076    0.0135    0.0297   0.0203  -0.0173   -0.0559   -0.0524    0.0199
  0.1445    0.2635    0.3125   0.2635   0.1445    0.0199   -0.0524   -0.0559
 -0.0173    0.0203    0.0297   0.0135  -0.0076   -0.016    -0.0093    0.0021
  0.0078    0.0055    0       -0.0031  -0.0024]
```

滤波器的幅度频率响应和相位频率响应如图 7.15 所示。

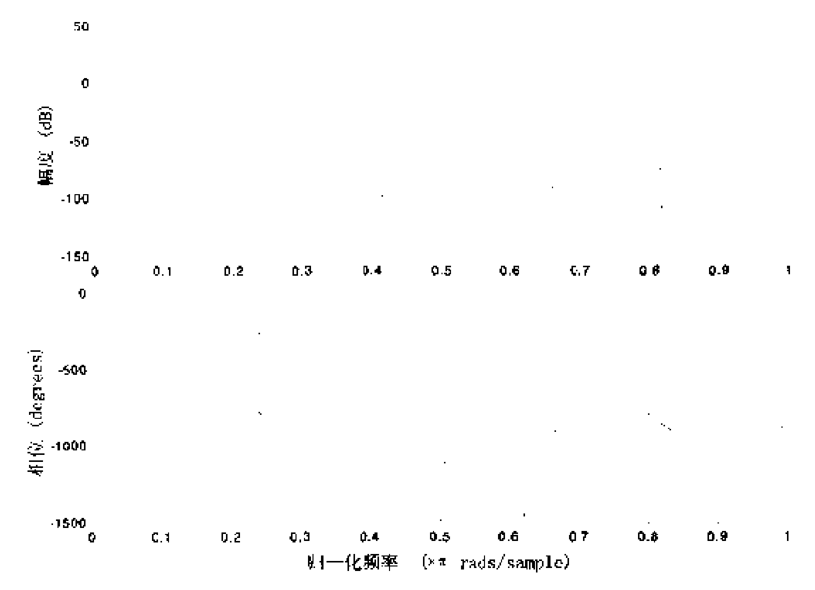


图7.15 利用凯塞窗设计的低通滤波器的频率响应

例，利用凯塞窗函数设计一个长度为奇数的带通滤波器(长度为奇数对应滤波器是偶阶的，由此能满足带通滤波器的要求)，通带范围是 1300Hz 到 2210Hz，阻带范围是 0Hz 到 1000Hz、2410Hz 到 4000Hz，阻带的波纹最大为 0.01，t 通带的波纹最大为 0.05，信号的采样频率为 8000Hz。

程序的清单如下：

```
fsamp = 8000;
fcuts = [1000 1300 2210 2410];
mags = [0 1 0];
devs = [0.01 0.05 0.01];
```

```
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,fsamp);
n = n + rem(n,2);
hh = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
[H,f] = freqz(hh,1,1024,fsamp);
plot(f,abs(H)),
grid on
```

得到滤波器的阶数为 90，滤波器的截止频率分别为 0.2875 和 0.5775，凯塞窗函数的 β 值为 3.3953。滤波器的幅度频率响应如图 7.16 所示。

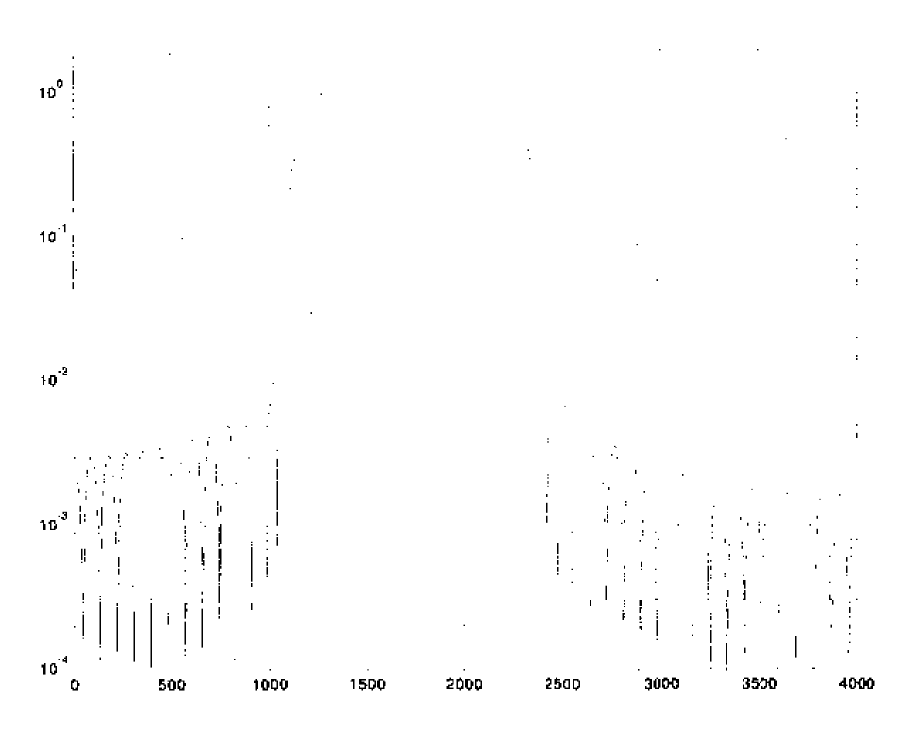


图7.16 滤波器的幅度频率响应图示

7.5.3 利用 fir2 设计任意响应 FIR 数字滤波器

所谓任意响应滤波器是指滤波器的幅度频率响应在指定的频段范围有不同的幅度值，例如要求滤波器在 0 到 $\pi/8$ 的幅度为 1，在 $\pi/8$ 到 $7\pi/8$ 的幅度为 1/2，在 $7\pi/8$ 到 π 的幅度响应为 1。对于设计任意响应的 FIR 数字滤波器，前面讨论的 fir1 函数是无能为力的，而 MATLAB 的工具箱中含有 fir2 能够完成这种滤波器的设计。fir2 函数的具体算法是：首先根据要求的幅度频率响应的向量形式进行插值，然后进行傅立叶反变换得到理想滤波器的单位脉冲响应，最后利用窗函数对理想滤波器的单位脉冲响应进行截短处理，由此得到 FIR 数字滤波器的系数。

fir2 函数的具体语法形式如下所示：

```
b = fir2(n,f,m)
b = fir2(n,f,m,window)
b = fir2(n,f,m,npt)
b = fir2(n,f,m,npt,window)
b = fir2(n,f,m,npt,lap)
```

```
b = fir2(n,f,m,npt,lap,window)
```

下面对函数的输入参数和返回值分别进行介绍：

- 输入参数 n 是待设计滤波器的阶数，在这儿值得注意的是，当设计的滤波器在频率为 π 的幅度响应不是 0 时，滤波器的阶数不能为奇数，因为对于线性相位滤波器的偶对称序列，在频率为 π 的幅度响应必然为 0。通常滤波器的阶数应该进行多次设计得到，或者通过 sptool 工具得到。
- 输入参数 f 是一个向量，它的元素为 0 到 1 的正数，对应为滤波器的频率，其中 0 对应于频率 0，而 1 对应于信号采样频率的一半。例如，要求设计的滤波器 0 到 π/8 的幅度为 1，在 π/8 到 2π/8 的幅度为 1/2，在 2π/8 到 4π/8 的幅度响应为 1/4，在 4π/8 到 6π/8 的幅度响应为 1/6，在 6π/8 到 π 的幅度响应为 1/8。则 f 可以表示为[0 0.125 0.125 0.250 0.250 0.500 0.500 0.750 0.750 1.00]。在 f 中的元素必须是单调递增的，其中相邻的两个元素可以相同，则对应于理想滤波器的幅度频率响应的不连续频率点。
- 输入参数 m 是一个向量，其中的元素是正实数，对应于 m 向量中频率点的幅度，在前面的举例中，m=[1 1 0.5 0.5 0.25 0.25 1/6 1/6 0.125 0.125]，向量 m 的长度和向量 f 相同。
- 输入参数 window 是一个向量，它是指窗函数的具体形式，它的长度为 n+1，当没有指定窗函数向量时，fir2 函数调用默认的窗函数向量为海明窗函数。
- 输入参数 npt 是一个正整数，它是在对幅度响应进行插值时的插值点的个数，默认值是 512。
- 输入参数 lap 是一个正整数，它是在对幅度响应进行插值时，对于不连续点转变成连续时的点数，默认值是 25。
- 函数的返回值 b 是设计出来的滤波器的系数组成的一个长度为 n+1 的向量。设计出来的滤波器的系统函数可以表示为：

$$B(z)=b(1)+b(2)z^{-1}+\cdots+b(n+1)z^{-n}$$

例，设计一个 60 阶的滤波器，要求设计的滤波器 0 到 π/8 的幅度为 1，在 π/8 到 2π/8 的幅度为 1/2，在 2π/8 到 4π/8 的幅度响应为 1/4，在 4π/8 到 6π/8 的幅度响应为 1/6，在 6π/8 到 π 的幅度响应为 1/8。并且画出理想滤波器和设计得到的滤波器的幅度频率响应进行比较。

程序清单如下：

```
f =[0 0.125 0.125 0.250 0.250 0.500 0.500 0.750 0.750 1.00];
m=[1 1 0.5 0.5 0.25 0.25 1/6 1/6 0.125 0.125];
b = fir2(60,f,m);
[h,w] = freqz(b);
plot(f,m,w/pi,abs(h))
grid on;
title(' 设计滤波器和理想滤波器幅度频率特性比较');
    xlabel('归一化频率(xfs)');
ylabel('幅度');
```

得到的频率响应如图 7.17 所示。

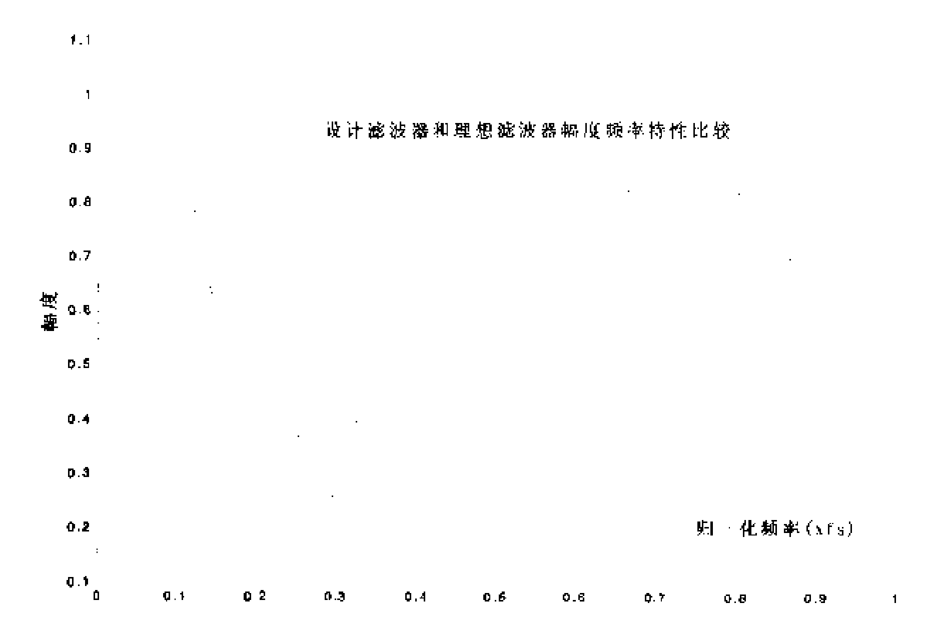


图7.17 利用fir2设计的任意频率响应的滤波器的频率响应

7.5.4 利用 remez 函数进行 FIR 滤波器的切比雪夫逼近法设计

在 7.4 节我们介绍了利用切比雪夫逼近法设计 FIR 数字滤波器的方法，和前面的方法相比较，切比雪夫逼近法有两个优点：它求得的滤波器是最优的；它可以根据设计的需要对不同频段进行加权处理。但是它的设计过程比较复杂，尤其是在最优化处理时，采用了雷米兹算法求得系统函数时比较繁琐。在 MATLAB 中有一个 remez 函数，利用这个函数可以很方便的设计出最优化的 FIR 数字滤波器，不必考虑其中繁琐的搜索过程。Remez 函数是采用了雷米兹算法，它的处理过程和前面讨论的雷米兹算法相同。

在 MATLAB 中利用 remez 函数的具体算法是：

```
b = remez(n,f,a)
b = remez(n,f,a,w)
b = remez(n,f,a,'ftype')
b = remez(n,f,a,w,'ftype')
b = remez(...,{lgrid})
b = remez(n,f,'fresp',w)
b = remez(n,f,'fresp',w,'ftype')
b = remez(n,f,{'fresp',p1,p2,...},w)
b = remez(n,f,{'fresp',p1,p2,...},w,'ftype')
[b,delta] = remez(...)
[b,delta,opt] = remez(...)
```

下面简单介绍 remez 函数的输入参数和返回值，对于函数语法后面的几个应用，由于篇幅所限，就不一一介绍。

- 输入参数 n 是待设计滤波器的阶数，在这儿值得注意的是，当设计的滤波器在频率为 π 的幅度响应不是 0 时，数波器的阶数不能为奇数，因为对于线性相位滤波器的偶对称序列，在频率为 π 的幅度响应必然为 0。通常滤波器的阶数应该进行

多次设计得到，或者通过 sptool 工具得到。

- 输入参数 f 是一个向量，它的元素为 0 到 1 的正数，对应为滤波器的频率，其中 0 对应于频率 0，而 1 对应于信号采样频率的一半。
- 输入参数 m 是一个向量，其中的元素是正实数，对应于 m 向量中频率点的幅度。
- 输入参数 w 是一个向量，它对应于各个频率段的加权值。
- 函数的返回值 b 是设计出来的滤波器的系数组成的一个长度为 n+1 的向量。设计出来的滤波器的系统函数可以表示为：

$$B(z) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

第8章 基于MATLAB的功率谱估计

功率谱估计涉及到信号与系统、随机信号分析、概率统计、随机过程及矩阵代数等一系列基础科学，广泛应用于雷达、声纳、通信、地质勘探、天文以及生物医学工程等众多领域，其内容、方法不断更新。

总的来讲，功率谱估计方法可以分成经典谱估计法与现代谱估计法。经典谱估计法又可分为直接法与间接法，直接法是利用快速傅立叶变换FFT算法对有限个样本数据进行傅氏变换得到功率谱的方法，又称为周期图法；间接法是先得到样本数据的自相关函数估计，然后进行傅氏变换得到功率谱的方法。由于直接法得到的功率谱估计存在谱曲线起伏大，或谱分辨率不高等缺点，又提出了几种改进算法，如Bartlett法、Welch法。

现代谱估计的提出主要是针对经典谱估计的分辨率低和方差性能不好等问题提出的。从现代谱估计的方法上，大致可分为参数模型谱估计和非参数模型谱估计。参数模型谱估计主要有AR模型、MA模型、ARMA模型等；非参数模型谱估计主要有最小方差方法以及MUSIC方法等。

在MATLAB 6中，有许多函数用来实现经典谱估计与现代谱估计，见表8.1。

表8.1 MATLAB中实现功率谱估计的函数

函数名	功能描述
Arburg	利用Burg算法估计AR模型参数
Arcov	利用协方差法估计AR模型参数
Armcov	利用改进协方差法估计AR模型参数
Aryule	利用Yule-Walker算法估计AR模型参数
Cohere	相关函数平方幅值估计
Corrcoef	计算相关系数
Csd	互谱密度估计
Levinson	Levinson-durbin递归算法
Lpc	线性预测系数
Pburg	利用Burg算法估计功率谱密度
Pcov	利用协方差法估计功率谱密度
Periodogram	利用周期图法估计功率谱密度
Pmcov	利用改进协方差法估计功率谱密度
Pmusic	利用MUSIC算法估计功率谱密度

续表

函数名	功能描述
Psdplot	绘制功率谱密度曲线
Pwelch	利用 Welch 算法估计功率谱密度
Pyulear	利用 Yule-Walker 算法估计功率谱密度
Xcorr	互相关函数估计

8.1 相关函数估计

广义平稳随机序列 $x(n)$ 与 $y(n)$ 的互相关函数定义为：

$$R_{xy}(m)=E[x(n)y^*(n+m)] \tag{8.1.1}$$

但实际上我们只能得到序列的有限长度，基于序列 $x(n)$ 与 $y(n)$ 的 N 个采样值的一个互相关函数估计公式为：

$$\hat{R}_{xy}(m)=\frac{1}{N-|m|}\sum_{n=0}^{N-|m|-1}x(n)y^*(n+m) \tag{8.1.2}$$

因为该估计公式得到的互相关函数为有偏估计，所以常采用下面的无偏估计公式：

$$\hat{R}_{xy}(m)=\frac{1}{N}\sum_{n=0}^{N-|m|-1}x(n)y^*(n+m) \tag{8.1.3}$$

同理，可以得到随机序列 $x(n)$ 自相关函数估计的两种形式为：

$$\begin{cases}\hat{R}_x(m)=\dfrac{1}{N-|m|}\displaystyle\sum_{n=0}^{N-|m|-1}x(n)x^*(n+m)\\ \hat{R}_x(m)=\dfrac{1}{N}\displaystyle\sum_{n=0}^{N-|m|-1}x(n)x^*(n+m)\end{cases} \tag{8.1.4}$$

8.1.1 自相关函数的快速计算

利用(8.1.4)式计算 $\hat{R}_x(m)$ 时，如果 m 和 N 都比较大，则需要的乘法次数过大，因此其应用受到了限制。这时，可以利用 FFT 来实现对 $\hat{R}_x(m)$ 的快速计算。

用 FFT 计算自相关函数的一般步骤：

(1) 对 $x(n)$ 补 N 个零，得 $x'(n)$，对 $x'(n)$ 做 DFT 得 $X'(k)$，k=0，1，…，$2N-1$；

(2) 求 $X'(k)$ 的幅平方，然后除以 N，得 $\frac{1}{N}|X'(k)|^2$；

(3) 对 $\frac{1}{N}|X'(k)|^2$ 做逆变换，得 $\hat{R}'_x(m)$。

$\hat{R}'_x(m)$ 并不简单地等于 $\hat{R}_x(m)$，而是等于将 $\hat{R}_x(m)$ 中 $-(N-1)\le m<0$ 的部分向右平移 $2N$ 点后形成的新序列。

在 MATLAB 中，函数 xcorr 用来进行自相关函数估计，且为基于上述 FFT 的快速算法，

其格式为：

C = XCORR(A, 'flag')

该函数返回长度为 $2N-1$ 的自相关序列，参数'flag'用来指定自相关函数估计所采用的计算公式：

- 若 flag 为 none，则计算序列的非归一化行相关：

$$\hat{R}_x(m)=\sum_{n=0}^{N-|m|-1}x(n)y^*(n+m)$$

- 若 flag 为 biased，则计算自相关函数的有偏估计：

$$\hat{R}_x(m)=\frac{1}{N-|m|}\sum_{n=0}^{N-|m|-1}x(n)x^*(n+m)$$

- 若 flag 为 unbiased，则计算自相关函数的无偏估计：

$$\hat{R}_x(m)=\frac{1}{N}\sum_{n=0}^{N-|m|-1}x(n)x^*(n+m)$$

- 若 flag 为 coeff，则对序列进行归一化处理，使得对零滞后的样本其自相关序列恒为 1。

例如下面的语句：

```
x=ones(1,7);
rx=xcorr(x,'none')
brx=xcorr(x,'biased')
ubrx=xcorr(x,'unbiased')
crx=xcorr(x,'coeff')
```

得到如下结果：

```
rx =
    1.0000    2.0000    3.0000    4.0000    5.0000    6.0000    7.0000
    6.0000    5.0000    4.0000    3.0000    2.0000    1.0000
brx =
    0.1429    0.2857    0.4286    0.5714    0.7143    0.8571    1.0000
    0.8571    0.7143    0.5714    0.4286    0.2857    0.1429
ubrx =
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000
crx =
    0.1429    0.2857    0.4286    0.5714    0.7143    0.8571    1.0000
    0.8571    0.7143    0.5714    0.4286    0.2857    0.1429
```

8.1.2 相关系数的计算

两个平稳随机序列 $x(n)$ 与 $y(n)$ 的相关系数矩阵估计为：

$$\hat{\rho}=\begin{bmatrix}\hat{\rho}_x & \hat{\rho}_{xy}\\ \hat{\rho}_{yx} & \hat{\rho}_y\end{bmatrix} \tag{8.1.5}$$

其中 $\hat{\rho}_x$ 、 $\hat{\rho}_y$ 分别为序列 $x(n)$ 与 $y(n)$ 的自相关系数，其值为 1； $\hat{\rho}_{xy}$ 、 $\hat{\rho}_{yx}$ 为序列 $x(n)$ 与 $y(n)$ 的互相关系数，定义为：

$$\hat{\rho}_{xy} = \sum_{n=0}^{N-1} x(n)y^*(n) \bigg/ \left[\sum_{n=0}^{N-1}|x(n)|^2 \sum_{n=0}^{N-1}|y(n)|^2\right]^{1/2}$$

$$\hat{\rho}_{yx} = \sum_{n=0}^{N-1} y(n)x^*(n) \bigg/ \left[\sum_{n=0}^{N-1}|x(n)|^2 \sum_{n=0}^{N-1}|y(n)|^2\right]^{1/2} \tag{8.1.6}$$

可见 $\hat{\rho}_{xy}$ 与 $\hat{\rho}_{yx}$ 互为共轭。

在 MATLAB 中，函数 corrcoef 用来计算两序列的相关系数矩阵，其格式为：

C=Corrcoef(x,y)

例如，下面的语句用来计算两有限长序列的相关系数矩阵：

```
x=randn(1,5);
y=linspace(0,1,5);
z=corrcoef(x,y)
```

得到的相关系数矩阵为：

```
z =
     1.0000    0.0998
     0.0998    1.0000
```

8.1.3 相干函数

序列 $x(n)$ 与 $y(n)$ 之间的相干函数定义为：

$$C_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_x(f)P_y(f)} \tag{8.1.7}$$

其中 $P_x(f)$ 、 $P_y(f)$ 分别为序列 $x(n)$ 与 $y(n)$ 的功率谱， $P_{xy}(f)$ 为它们的互功率谱，则相干函数反映了两序列功率谱之间的关系，其值在 0~1 之间。

在 MATLAB 中，函数 cohere 采用 FFT 算法来计算两序列的相干函数，其基本格式为：

```
[Cxy,F] = COHERE(x,y,NFFT,Fs,WINDOW,NOVERLAP,DFLAG)
```

参数说明如下：

- x、y 为两有限长序列。
- NFFT 指定 FFT 算法的长度，其默认值为 NFFT=min(256,length(x))，为了提高计算速度，常取 NFFT 为 2 的整次幂。
- Fs 为采样频率，其默认值为 2。
- WINDOW 指定加窗函数，默认是取 WINDOW=hanning(nfft)，其他的窗函数如海明窗、布拉克曼窗、巴特里特窗及凯塞窗等。
- NOVERLAP 指定分段重叠的样本数，这个参数的用法下面还做详细介绍，其默认值为 0。

- DFLAG 用于指定处理方式：
  当 DFLAG= 'linear'时，可从预加权的 x、y 段中删去最佳的直线拟合；
  当 DFLAG= 'mean'时，可从预加权的 x、y 段中删去均值；
  当 DFLAG='none'时，不做任何处理。
- F 得到的估计相关性的频率点。
- Cxy 为输出的相干函数。当 x、y 为实数时，只估计正频率处的相干函数，若 NFFT 为偶数，则 Cxy 为 NFFT/2+1 维的列矢量，若 NFFT 为奇数，则 Cxy 为(NFFT+1)/2 维的列矢量；当 x 或 y 为复数时，估计正负频率处的相干函数，Cxy 的长度为 NFFT。
- 若输出参数默认，则可在当前图形中绘制相干估计的频域曲线。

例如，输入下面的语句：

```
b=firl(20,.3,blackman(21));
bl=ones(1,20)/sqrt(20);
s=randn(10000,1);
x=filter(bl,1,s);
y=filter(b,1,s);
cohere(x,y,1024,2,[],512,'linear')
```

得到如图 8.1 所示的相干函数曲线。

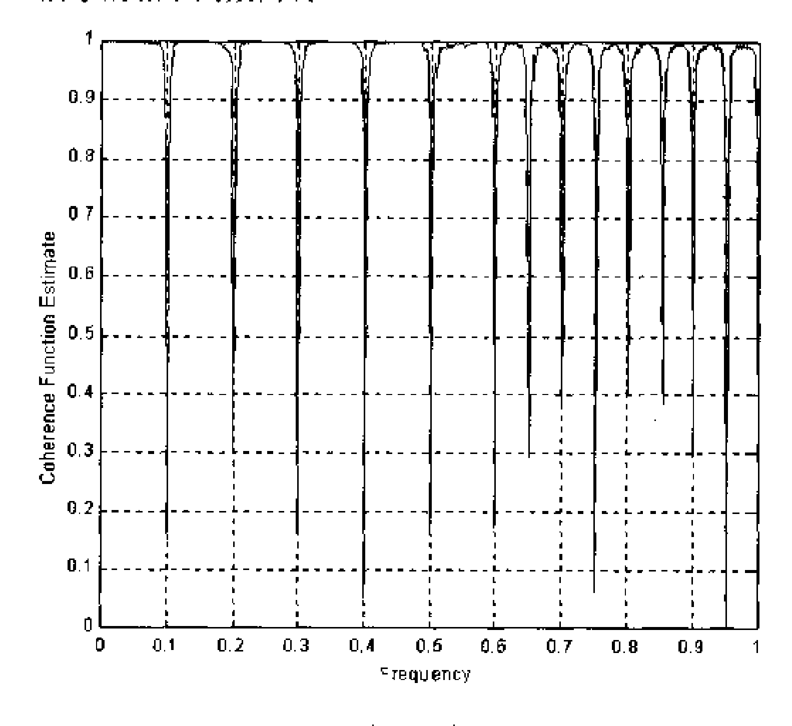


图8.1 相干函数曲线

8.2 经典谱估计方法

本节重点讨论了经典谱估计的两个主要方法：直接法与间接法，最后给出了两种改进的直接法：Bartlett 法与 Welch 法。

8.2.1 直接法

直接法又称周期图法，它是把随机序列 $x(n)$ 的 N 个观测数据视为一能量有限的序列，直接计算 $x(n)$ 的离散傅立叶变换，得 $X(k)$，然后再取其幅值的平方，并除以 N，作为序列 $x(n)$ 真实功率谱的估计，即：

$$\hat{P}_{\mathrm{PER}}(k)=\frac{1}{N}|X(k)|^2 \tag{8.2.1}$$

例如，下面的语句，即利用直接法计算一含噪序列的功率谱：

```
%figure 8.2
Fs=1000;%采样频率
%产生含有噪声的序列
n=0:1/Fs:1;
xn=cos(2*pi*40*n)+3*cos(2*pi*100*n)+randn(size(n));
%计算序列的 DFT
nfft=1024;
Xk=fft(xn,nfft);
%计算序列的 PSD
Pxx=abs(Xk).^2/length(n);
%绘制图形
index=0:round(nfft/2-1);
k=index*Fs/nfft;
plot_Pxx=10*log10(Pxx(index+1));
plot(k,plot_Pxx)
```

得到如图 8.2 所示的图形。

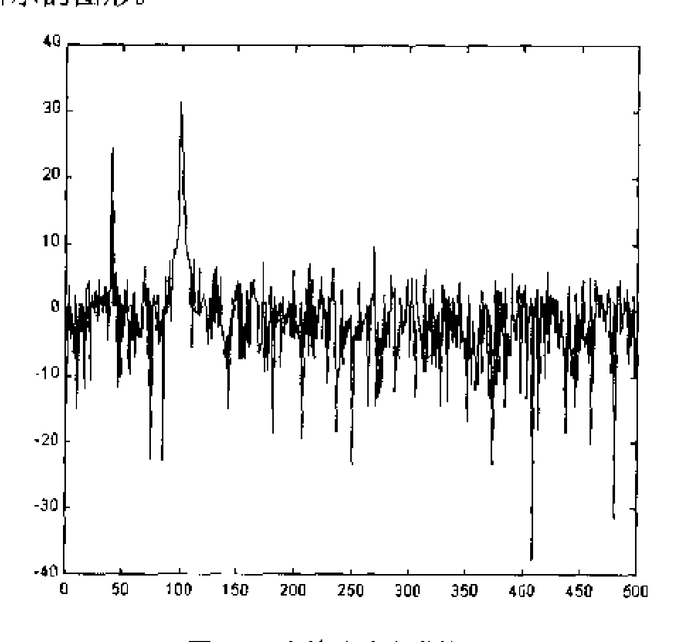


图8.2 直接法功率谱估计

另外，在 MATLAB 中，函数 periodogram 也可实现周期图法功率谱估计，其格式为：

[Pxx,F] = PERIODOGRAM(x,WINDOW,NFFT,Fs)

参数说明：

- x 为进行功率谱估计的输入有限长序列。
- WINDOW 用于指定采用的窗函数，默认时采用矩形窗，窗长等于输入序列 x 的长度。对于周期图法，算法未指定窗函数，但有限长序列相当于采用矩形窗。
- NFFT 设定 FFT 算法的长度，一般取 2 的整次幂，默认值为 256。
- Fs 为采样频率，默认值为 1。
- Pxx 为输出的功率谱估计值。
- F 为得到的频率点。

例如，对于上面的例子，采用函数 periodogram 实现，得到如图 8.3 所示的功率谱曲线。

```
%figure 8.3
Fs=1000;%采样频率
%产生含有噪声的序列
n=0:1/Fs:1;
xn=cos(2*pi*40*n)+3*cos(2*pi*100*n)+randn(size(n));
%参数设置并计算序列的 PSD
window=boxcar(length(xn));
nfft=1024;
[Pxx,f]=periodogram(xn,window,nfft,Fs);
%绘制图形
plot(f,10*log10(Pxx))
```

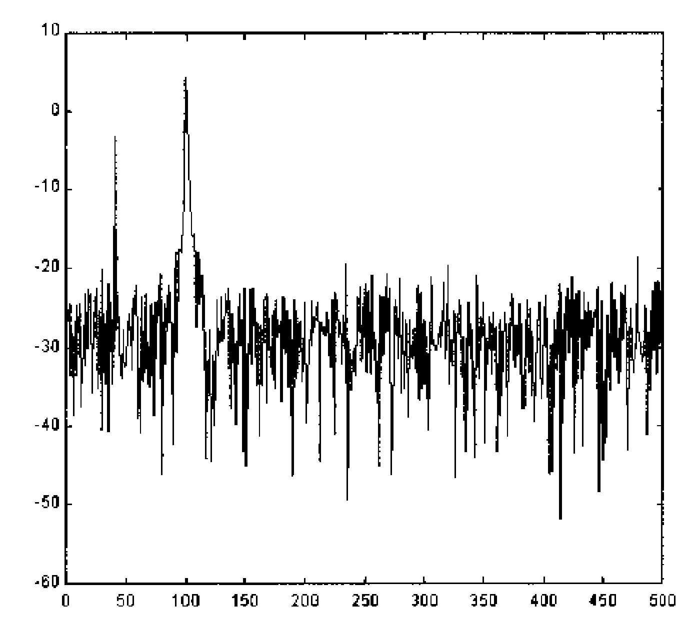


图8.3 利用函数periodogram得到的功率谱估计

8.2.2 间接法

间接法先由序列 $x(n)$ 估计出自相关函数 $\hat{R}_x(m)$，然后对 $\hat{R}_x(m)$ 求傅立叶变换，便得到 $x(n)$ 的功率谱估计，即：

$$\hat{P}_{\mathrm{BT}}(k)=\sum_{m=-N}^{N}\hat{R}_x(m)W_N^{-mk} \tag{8.2.2}$$

例如，对于上面的例子，若采用间接法，可以通过下面的程序实现：

```
%figure 8.4
Fs=1000;%采样频率
%产生含有噪声的序列
n=0:1/Fs:1;
xn=cos(2*pi*40*n)+3*cos(2*pi*100*n)+randn(size(n));
%计算序列的自相关函数
cxn=xcorr(xn,'unbiased');
%求序列的 PSD
nfft=1024;
CXk=fft(cxn,nfft);
Pxx=abs(CXk);
%绘制图形
index=0:round(nfft/2-1);
k=index*Fs/nfft;
plot_Pxx=10*log10(Pxx(index+1));
plot(k,plot_Pxx)
```

得到的功率谱曲线如图 8.4 所示。

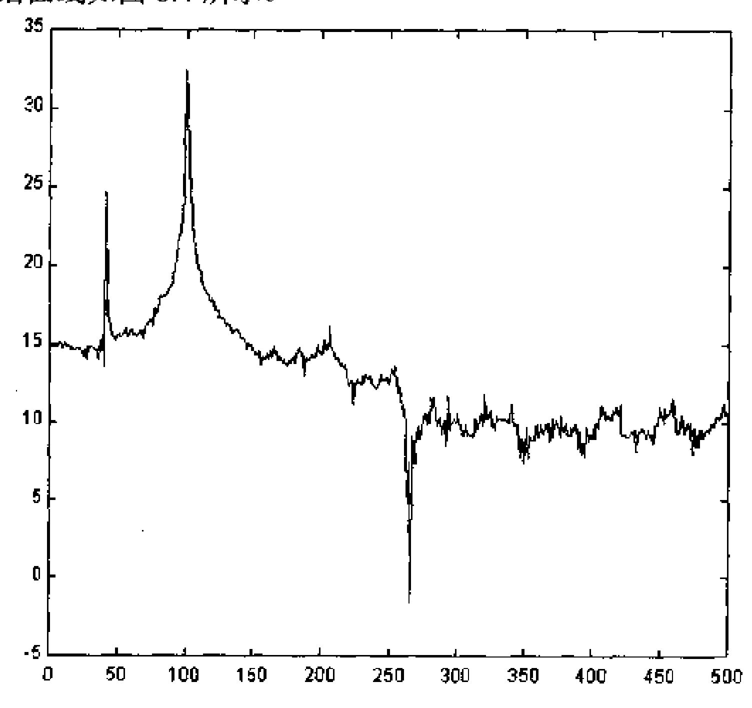


图8.4　间接法功率谱估计

8.2.3　改进的直接法

对于直接法的功率谱估计，当数据长度 N 太大时，谱曲线起伏加剧，若 N 太小，谱的分辨率又不好，因此需要改进。

1. Bartlett法

Bartlett 平均周期图的方法是将 N 点的有限长序列 $x(n)$ 分段求周期图再平均。

设将 $x(n)$ 分成 L 段，每段有 M 个样本，因而 $N=LM$，第 i 段样本序列可写成：

$$x^i(n)=x(n+iM-M) \qquad 0\leqslant n\leqslant M-1,\ 1\leqslant i\leqslant L$$

第 i 段的周期图为：

$$\hat{P}_{\mathrm{PER}}^{i}(k)=\frac{1}{M}\left|\sum_{n=0}^{M-1}x^i(n)W_M^{-kn}\right|^2 \tag{8.2.3}$$

如果 $m>M$，$\hat{R}_x(m)$ 很小，则可假定各段的周期图是互相独立的，则谱估计可定义为 L 段周期图的平均，即：

$$\hat{P}_{\mathrm{PER}}(k)=\frac{1}{L}\sum\hat{P}_{\mathrm{PER}}^{i}(k) \tag{8.2.4}$$

在 MATLAB 中，可以利用函数 psd 来实现 Bartlett 平均周期图方法的功率谱估计，其格式为：

```
[Pxx, Pxxc, F] = PSD(x,NFFT,Fs,WINDOW,NOVERLAP,P)
```

其中，参数 Pxx、F、x、NFFT、Fs、WINDOW、NOVERLAP 的说明可参照 8.1 节的 cohere 函数，但对 Bartlett 平均周期图方法来说，需要做以下几点说明：

- 因为 Bartlett 方法中没有指定窗函数，但对于有限长序列来说，相当于采用矩形窗，即参数 WINDOW 应设为 boxcar 窗。
- Bartlett 方法中没有指定数据重叠，因而参数 NOVERLAP 应设为 0。
- 利用 psd 函数实现 Bartlett 方法，其分段数 L 默认为：

  $$\mathrm{L=fix}\left(\frac{\text{数据长度}}{\text{窗长度}}\right)$$

  其中 fix 表示数值朝零方向取整。
- 参数 P 为置信概率，参数 Pxxc 为 Pxx 的 P×100%置信区间估计。

例如，对于上面的例子，采用 Bartlett 平均周期图方法估计序列的功率谱，其实现程序为：

```
%figure 8.5
Fs=1000;%采样频率
%产生含有噪声的序列
n=0:1/Fs:1;
xn=cos(2*pi*40*n)+3*cos(2*pi*100*n)+randn(size(n));
%参数设置
nfft=1024;
window=boxcar(1001);
noverlap=0;
p=0.9;
%计算序列的 PSD
[Pxx,Pxxc]=psd(xn,nfft,Fs,window,noverlap,p);
%绘制图形
```

```
index=0:round(nfft/2-1);
k=index*Fs/nfft;
plot_Pxx=10*log10(Pxx(index+1));
plot_Pxxc=10*log10(Pxxc(index+1));
%功率谱的对数曲线
figure(1)
plot(k,plot_Pxx)
%置信区间中的功率谱曲线
figure(2)
plot(k,[plot_Pxx plot_Pxx-plot_Pxxc plot_Pxx+plot_Pxxc])
```

得到的曲线如图 8.5 与 8.6 所示。

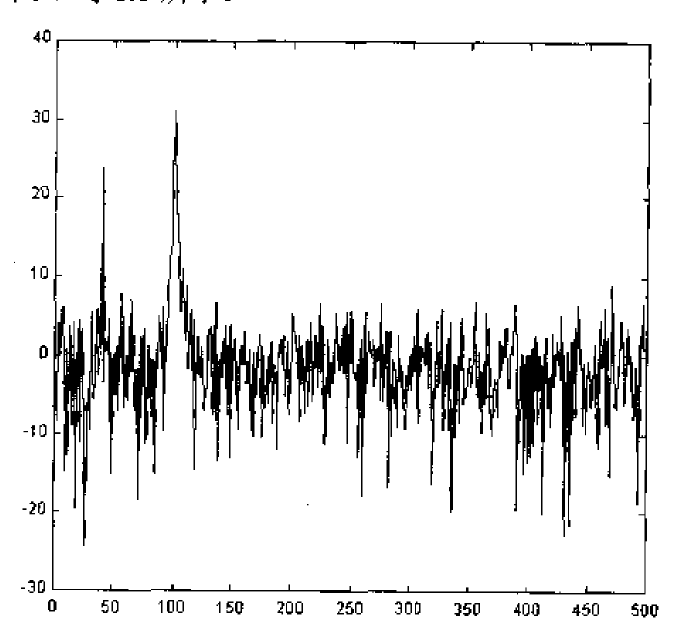


图8.5 Bartlett法得到的功率谱曲线

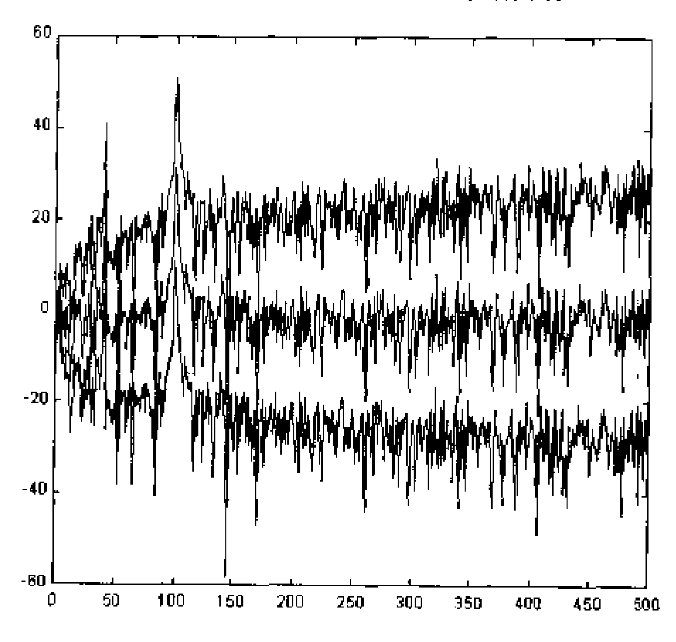


图8.6 Bartlett法得到的置信区间功率谱曲线

2. Welch法

Welch 法对 Bartlett 法进行了两方面的修正，一是选择适当的窗函数 $w(n)$，并在周期图计算前直接加进去，这样得到的每一段的周期图为：

$$\hat{P}_{\mathrm{PER}}^{i}(k)=\frac{1}{MU}\left|\sum_{n=0}^{M-1}x^{i}(n)w(n)W_{M}^{-kn}\right|^{2} \tag{8.2.6}$$

这里 $U=\frac{1}{M}\sum_{n=0}^{M-1}w^{2}(n)$ 为归一化因子，加窗的优点是无论什么样的窗函数均可使谱估计非负。二是在分段时，可使各段之间有重叠，这样会使方差减小。

在 MATLAB 中，函数 psd 与 pwelch 都可实现 Welch 法的功率谱估计，其方法是一样的，只是参数设置有所不同，这里以函数 pwelch 为例进行说明。

函数 pwelch 的格式为：

```
[Pxx,Pxxc,F]=PWELCH(x,NFFT,Fs,WINDOW,NOVERLAP,P,RANGE,MAGUNITS)
```

参数说明：

- 参数 x、NFFT、Fs、WINDOW、NOVERLAP、P、Pxx、Pxxc、F 的说明可参照函数 psd，只是可以指定参数 WINDOW 为各种窗函数，参数 NOVERLAP 可以为非零值。
- 在函数 pwelch 中，分段数 L 为：
$$\mathrm{L=fix}\left(\frac{\text{数据长度}-\mathrm{NOVERLAP}\text{值}}{\text{窗长度}-\mathrm{NOVERLAP}\text{值}}\right)$$
- 参数 RANGE 用来指定频率间隔：
  若 RANGE= 'half'，频率间隔为[0,Fs/2]；
  若 RANGE= 'whole'，频率间隔为[0,Fs]。
- 参数 MAGUNITS 用来指定绘图格式：
  若 MAGUNITS='squared'，采用一般绘图格式；
  若 MAGUNITS= 'db'，采用分贝绘图格式。

例如，再以前面的例子为例，采用 Welch 法计算序列的功率谱，实现程序如下：

```
%figure 8.7
Fs=1000;
n=0:1/Fs:1;
xn=cos(2*pi*40*n)+3*cos(2*pi*100*n)+randn(size(n));
nfft=1024;
window=boxcar(100);
noverlap=20;
p=0.9;
range='half';
magunits='square';
[Pxx,f]=pwelch(xn,nfft,Fs,window,noverlap,p,range,magunits);
plot_Pxx=10*log10(Pxx);
plot(f,plot_Pxx)
```

得到如图 8.7 所示的功率谱曲线。

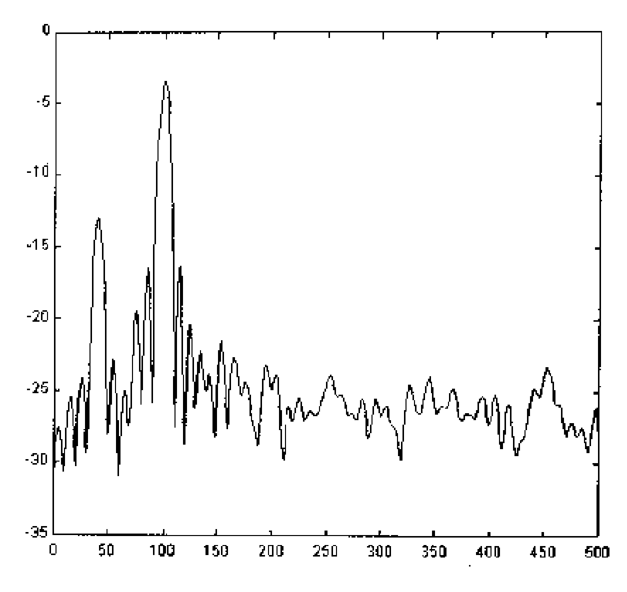


图8.7 Welch法得到的功率谱曲线(矩形窗)

比较图 8.7 与图 8.5 所示的曲线，从中可以看出，虽然 Welch 法和 Bartlett 法都采用了矩形窗，但由于 Welch 法在分段时，可使各段之间有重叠，而 Bartlett 法则没有，所以 Welch 法得到的曲线要比 Bartlett 法得到的曲线误差要小。

如果将上面程序中的窗函数改为采用海明窗或布莱克曼窗，即函数 pwelch 中的参数 WINDOW 如下设置：

```
window=hamming(100);
```

或

```
window=blackman(100);
```

得到的功率谱曲线分别如图 8.8 与图 8.9 所示。

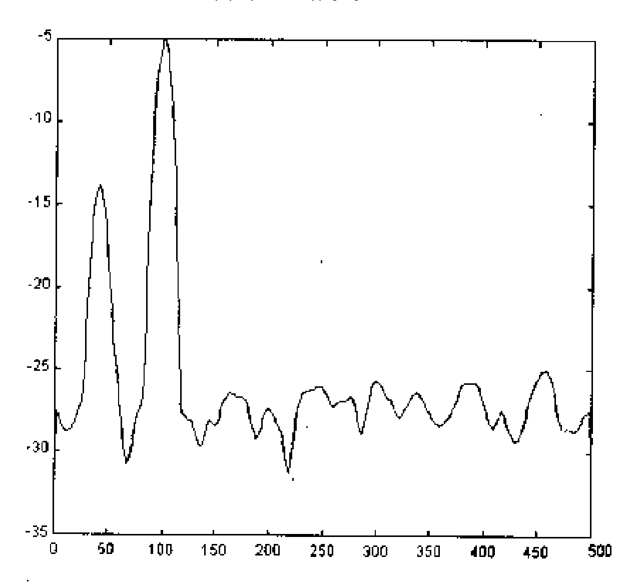


图8.8 Welch法得到的功率谱曲线(Hamming窗)

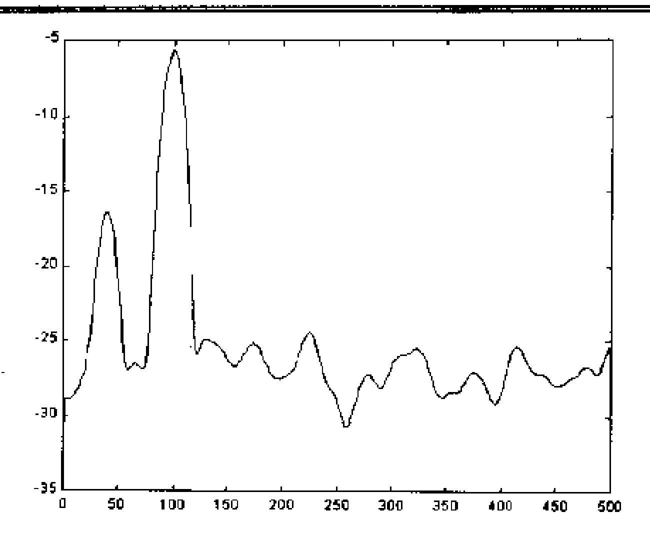


图8.9 Welch法得到的功率谱曲线(Blackman窗)

8.3 AR 模型功率谱估计

AR 模型功率谱估计是现代谱估计的主要内容，本节首先给出了AR模型的 Yule-Walker 方程，然后介绍了 Levinson-Durbin 递推算法与 AR 模型参数的其它求解算法，并讨论了有关 AR 模型阶数 p 的选择问题，最后讲述了 MATLAB 中 AR 模型谱估计的函数及 AR 模型谱估计的性质。

8.3.1 AR 模型的 Yule-Walker 方程

AR 模型又成为自回归模型，它是一个全极点的模型，该模型现在输出是现在的输入和过去输出的加权和，可用如下差分方程来表示：

$$x(n)=-\sum_{r=1}^{p}a_r x(n-k)+u(n) \tag{8.3.1}$$

其中 $u(n)$ 为白噪声序列， p 为 AR 模型的阶数， a_r $r=1,2,\cdots,p$ 为 AR 模型的参数。

由上面的差分方程，我们很容易得到 AR 模型的转移函数形式：

$$H(z)=\frac{1}{1+\sum_{r=1}^{p}a_k z^{-r}} \tag{8.3.2}$$

进一步可以得到利用 AR 模型进行功率谱估计的公式：

$$\hat{P}_x(k)=\frac{\sigma^2}{\left|1+\sum_{r=1}^{p}a_r W_N^{-kr}\right|^2} \tag{8.3.3}$$

其中 σ^2 为白噪声序列的方差。

由此可以看出，要进行功率谱估计，必须求得 AR 模型的参数 a_1， a_2， $\cdots$， a_p 及 σ^2，它们可由下面的 Yule-Walker 方程求得：

$$\begin{bmatrix} \hat{R}_x(0) & \hat{R}_x(-1) & \cdots & \hat{R}_x(-p) \\ \hat{R}_x(1) & \hat{R}_x(0) & \cdots & \hat{R}_x(-(p-1)) \\ \vdots & \vdots & \cdots & \vdots \\ \hat{R}_x(p) & \hat{R}_x(p-1) & \cdots & \hat{R}_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.3.4}$$

8.3.2 Levinson-Durbin 递推算法

利用 AR 模型进行功率谱估计，必须计算出 AR 模型的参数 a_1，a_2，…，a_p 及白噪声序列的方差 σ^2，即必须求解 Yule-Walker 方程，这可用 Levinson-Durbin 递推算法实现。

Levinson-Durbin 递推算法的递推公式为：

$$a_{rr} = -[\hat{R}_x(r) + \sum_{l=1}^{r-1} a_{r-1,l}\hat{R}_x(r-l)]/\sigma_{r-1}^2 \tag{8.3.5}$$

$$a_{ri} = a_{r-1,i} + a_{r,r}a_{r-1,r-i} \tag{8.3.6}$$

$$\sigma_r^2 = (1-|a_{rr}|^2)\sigma_{r-1}^2, \quad \sigma_0^2 = \hat{R}_x(0) \tag{8.3.7}$$

在 MATLAB 中，函数 levinson 与 aryule 都采用 Levinson-Durbin 递推算法来求解 AR 模型的参数 a_1，a_2，…，a_p 及白噪声序列的方差 σ^2，只是二者的输入参数不同，它们的格式为：

```
A = LEVINSON(R, ORDER)
A = ARYULE(x,ORDER)
```

两函数均为定阶 ORDER 的求解，但函数 levinson 的输入参数要求是序列的自相关函数，而函数 aryule 的输入参数为采样序列。

下面语句说明函数 levinson 与 aryule 的功能是相同的。

```
randn('seed',0)
a=[1 0.1 0.2 0.3 0.4 0.5];
x=impz(1,a,20)+randn(20,1)/20;
r=xcorr(x,'biased');
r(1:length(x)-1)=[];
A=levinson(r,5)
B=aryule(x,5)
```

得到结果为：

```
A =
    1.0000    0.0298    0.1985    0.2681    0.3866    0.3502
B =
    1.0000    0.0298    0.1985    0.2681    0.3866    0.3502
```

8.3.3 AR 模型参数的其他求解算法

令 $b_p(n)$ 与 $e_p(n)$ 分别为线性预测 AR 模型的后向预测误差和前向预测误差：

$$b_p(n) = a_{p0}x(n-p) + a_{p1}x[n-(p-1)] + \cdots + a_{pp}x(n), \quad a_{p0} = 1 \tag{8.3.8}$$

$$e_p(n)=a_{p0}x(n)+a_{p1}x(n-1)+\cdots+a_{pp}x(n-p)\,,\quad a_{p0}=1 \tag{8.3.9}$$

P_b、P_e分别为后向预测误差功率与前向预测误差功率：

$$P_b=[b_p]^{\mathrm{H}}b_p\,,\quad P_e=[e_p]^{\mathrm{H}}e_p \tag{8.3.10}$$

1. Burg算法

令前后向预测误差功率之和为：

$$P_{eb}=\frac{1}{2}[P_e+P_b] \tag{8.3.11}$$

其中：

$$P_e=\frac{1}{N-p}\sum_{n=p}^{N-1}|e_p(n)|^2 \tag{8.3.12}$$

$$P_b=\frac{1}{N-p}\sum_{n=p}^{N-1}|b_p(n)|^2 \tag{8.3.13}$$

当阶次r由1至p时，$b_p(n)$与$e_p(n)$有如下的递推关系：

$$\begin{aligned} e_r(n)&=e_{r-1}(n)+k_rb_{r-1}(n-1)\\ b_r(n)&=b_{r-1}(n-1)+k_re_{r-1}(n) \end{aligned} \tag{8.3.14}$$

且 $e_0(n)=b_0(n)=x(n)$

式中k_r为反射系数，且$k_r=a_{rr}$。

Burg 算法就是使得前后向预测误差功率之和P_{eb}相对于反射系数k_r最小，由此可以求得k_r的估计公式：

$$\hat{k}_r=\frac{-2\sum_{n=r}^{N-1}e_{r-1}(n)b_{r-1}(n-1)}{\sum_{n=r}^{N-1}|e_{r-1}(n)|^2+\sum_{n=r}^{N-1}|b_{r-1}(n-1)|^2} \tag{8.3.15}$$

由上式估计出$\hat{k}_r$后，在阶次r时的 AR 模型参数仍然由 Levinson-Durbin 递推算法求出：

$$\begin{aligned} a_{ri}&=a_{r-1,i}+\hat{k}_ra_{r-1,r-i}\\ a_{rr}&=\hat{k}_r\\ \sigma_r^2&=(1-|\hat{k}_r|^2)\sigma_{r-1}^2\,,\quad \sigma_0^2=\hat{R}_x(0)=\frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2 \end{aligned} \tag{8.3.16}$$

在 MATLAB 中，函数 arburg 就是利用上述的 Burg 算法计算 AR 模型的参数，其格式为：

```
A = ARBURG(x,ORDER)
```

其中 x 为有限长序列，参数 ORDER 用来指定 AR 模型的阶数。

例如，以上面的例子为例，利用 Burg 算法估计 AR 模型的参数，其实现程序如下：

```
randn('seed',0)
a=[1 0.1 0.2 0.3 0.4 0.5];
x=impz(1,a,20)+randn(20,1)/20;
```

```
A=arburg(x,5)
```

得到如下结果：

```
A =
    1.0000    0.1642    0.4595    0.3636    0.4658    0.5993
```

2. 改进的协方差方法

如同 Burg 算法一样，仍是令前后向预测误差功率之和为：

$$P_{eb}=\frac{1}{2}[P_e+P_b]$$

式中：

$$P_e=\frac{1}{N-p}\sum_{n=p}^{N-1}|e_p(n)|^2$$

$$P_b=\frac{1}{N-p}\sum_{n=p}^{N-1}|b_p(n)|^2$$

与 Burg 算法不同的是，改进的协方差方法不是仅使得前后向预测误差功率之和 P_{eb} 相对于反射系数 k_r 最小，而是使得 P_{eb} 相对 $a_{ri}\,(i=1,2,\cdots,r)$ 都为最小，r 由 1 到 p。由此得到改进的协方差方法的正则方程为：

$$\begin{bmatrix}c_x(1,1) & c_x(1,2) & \cdots & c_x(1,p)\\ c_x(2,1) & c_x(2,2) & \cdots & c_x(2,p)\\ \vdots & \vdots & \cdots & \vdots\\ c_x(p,1) & c_x(p,2) & \cdots & c_x(p,p)\end{bmatrix}\begin{bmatrix}a_1\\ a_2\\ \vdots\\ a_p\end{bmatrix}=-\begin{bmatrix}c_x(1,0)\\ c_x(2,0)\\ \vdots\\ c_x(p,0)\end{bmatrix} \tag{8.3.17}$$

与

$$\sigma^2=c_x(0,0)+\sum_{r=1}^{p}a_r c_x(0,r) \tag{8.3.18}$$

式中：

$$c_x(i,k)=\frac{1}{2(N-p)}[\sum_{n=p}^{N-1}x(n-i)x(n-k)+\sum_{n=0}^{N-1-p}x(n+k)x(n+i)] \tag{8.3.19}$$

在 MATLAB 中，可以通过函数 armcov 来实现上面的改进协方差算法，函数格式如下：

```
A = ARMCOV(x,ORDER)
```

该函数用来计算有限长序列 x(n)的 ORDER 阶 AR 模型参数。

例如，输入下面的语句：

```
randn('seed',0)
a=[1 0.1 0.2 0.3 0.4 0.5];
x=impz(1,a,20)+randn(20,1)/20;
A=armcov(x,5)
```

得到的 AR 模型参数为：

```
A =
    1.0000    0.2761    0.3838    0.0511    0.8198    0.5885
```

8.3.4 AR 模型阶数 p 的选择

AR 模型的阶数 P 一般事先是不知道的，需要事先选定一个稍大的值，在递推的过程中确定。在使用 Levinson-Durbin 递推算法时，可以给出由低阶到高阶的每一组参数，且模型的最小预测误差功率 $P_{\min}$（相当于白噪声序列的方差 σ^2）是递减的。直观上讲，当预测误差功率 P 达到指定的希望值，或是不再发生变化时，这时的阶数即是应选的正确阶数。

因为预测误差功率 P 是单调下降的，因此，该值降到多少才合适，往往不好选择。为此，有几个不同的准则被提出，其中较常见的两个是：

- 最终预测误差准则

$$\mathrm{FPE}(r)=P_r\frac{N+(r+1)}{N-(r+1)} \tag{8.3.20}$$

- 信息论准则

$$\mathrm{AIC}(r)=N\ln(P_r)+2r \tag{8.3.21}$$

式中 N 为有限长序列 $x(n)$ 的长度，当阶数 r 由 1 增加时，FPE(r) 和 AIC(r) 都将在某一个 r 处取得极小值。将此时的 r 定为最合适的阶数 p。在实际运用时发现，当数据较短时，它们给出的阶次偏低，且二者给出的结果基本上是一致的。

应该指出，上面两式仅为阶数选择提供了一个依据，对所研究的某一个具体信号，究竟阶数取多少为最好，还要在实践中由所得的结果作多次比较后，予以确定，8.3.6 节将通过具体的例子来说明阶数对功率谱估计的影响。

8.3.5 MATLAB 中 AR 模型谱估计的函数说明

1. Pyulear函数

功能：利用 Yule-Walker 方法进行功率谱估计。

格式：

```
Pxx = Pyulear (x,ORDER,NFFT)
[Pxx,W] = Pyulear (x,ORDER,NFFT)
[Pxx,F] = Pyulear (x,ORDER,NFFT,Fs)
Pyulear (x,ORDER,NFFT,Fs, RANGE,MAGUNITS)
```

说明：

Pxx = Pyulear (x,ORDER,NFFT)中，采用 Yule-Walker 方法估计序列 x 的功率谱，参数 ORDER 用来指定 AR 模型的阶数，NFFT 为 FFT 算法的长度，其默认值为 256，若 NFFT 为偶数，则 Pxx 为 NFFT/2+1 维的列矢量，若 NFFT 为奇数，则 Pxx 为(NFFT+1)/2 维的列矢量；当 x 为复数时，Pxx 的长度为 NFFT。

[Pxx,W] = Pyulear (x,ORDER,NFFT)中，返回一个频率向量 W。

[Pxx,F] = Pyulear (x,ORDER,NFFT,Fs)中，可在 F 向量得到功率谱估计的频率点，Fs 指定采样频率。

Pyulear (x,ORDER,NFFT,Fs, RANGE,MAGUNITS)直接画出功率谱估计的曲线图，参数 RANGE

与 MAGUNITS 的说明可参照函数 pwelch。

2. Pburg函数

功能：利用 Burg 方法进行功率谱估计。

格式：

```
Pxx = Pburg(x,ORDER,NFFT)
[Pxx,W] = Pburg (x,ORDER,NFFT)
[Pxx,F] = Pburg (x,ORDER,NFFT,Fs)
Pburg (x,ORDER,NFFT,Fs, RANGE,MAGUNITS)
```

说明：Pburg 函数与 Pyulear 函数格式相同，只是计算 AR 模型参数时所采用的方法不同，因而其格式说明可参照 Pyulear 函数。

3. Pcov函数

功能：利用协方差方法进行功率谱估计。

格式：

```
Pxx = Pcov(x,ORDER,NFFT)
[Pxx,W] = Pcov (x,ORDER,NFFT)
[Pxx,F] = Pcov (x,ORDER,NFFT,Fs)
Pcov (x,ORDER,NFFT,Fs, RANGE,MAGUNITS)
```

说明：

Pcov 函数采用协方差方法估计 AR 模型的参数，然后计算序列 x 的功率谱。协方差方法与改进的协方差方法相比，前者是仅令前向预测误差为最小，其他步骤是一样的。由于其与 Pyulear 函数格式相同，只是计算 AR 模型参数时所采用的方法不同，因而其格式说明可参照 Pyulear 函数。

4. Pmcov函数

功能：利用改进协方差方法进行功率谱估计。

格式：

```
Pxx = Pmcov(x,ORDER,NFFT)
[Pxx,W] = Pmcov (x,ORDER,NFFT)
[Pxx,F] = Pmcov (x,ORDER,NFFT,Fs)
Pmcov (x,ORDER,NFFT,Fs, RANGE,MAGUNITS)
```

说明：

Pmcov 函数采用改进协方差方法估计 AR 模型的参数，然后计算序列 x 的功率谱，其格式说明可参照 Pyulear 函数。

下面通过一例子说明各函数的用法，并比较它们的区别。

例如，输入下面语句：

```
%figure 8.10-13
Fs=1000;%采样频率
%产生序列
```

```
n=0:1/Fs:.3;
xn=cos(2*pi*n*200)+randn(size(n));
%设置参数
order=20;
nfft=1024;
%Yule-Walker 方法
figure(1)
pyulear(xn,order,nfft,Fs,'half','db');
%Burg 方法
figure(2)
pburg(xn,order,nfft,Fs,'half','db');
%协方差方法
figure(3)
pcov(xn,order,nfft,Fs,'half','db');
%改进协方差方法
figure(4)
pmcov(xn,order,nfft,Fs,'half','db');
```

得到的功率谱估计曲线如图 8.10~图 8.13 所示。

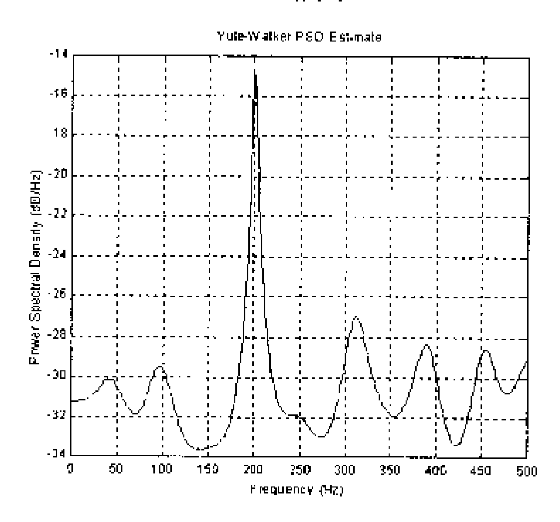


图8.10 Yule-Walker法得到的功率谱估计曲线

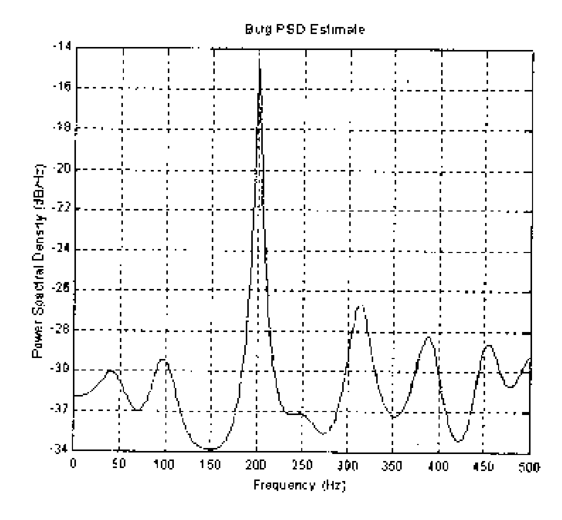


图8.11 Burg法得到的功率谱估计曲线

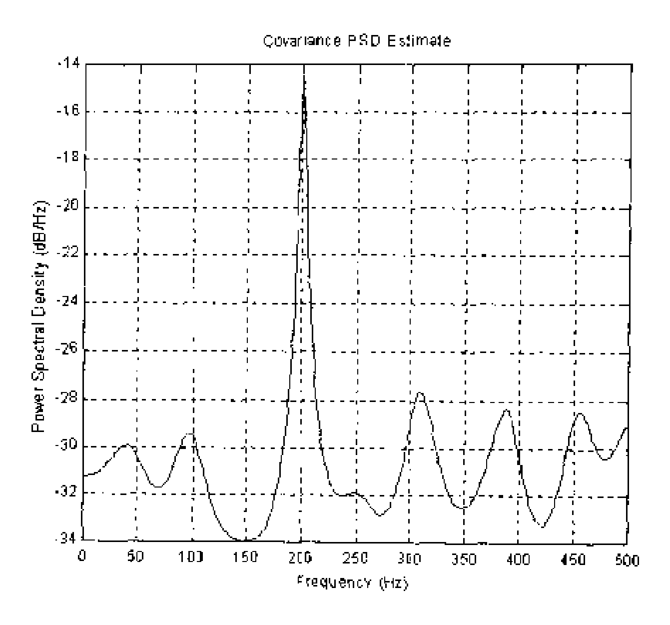


图8.12　协方差方法得到的功率谱估计曲线

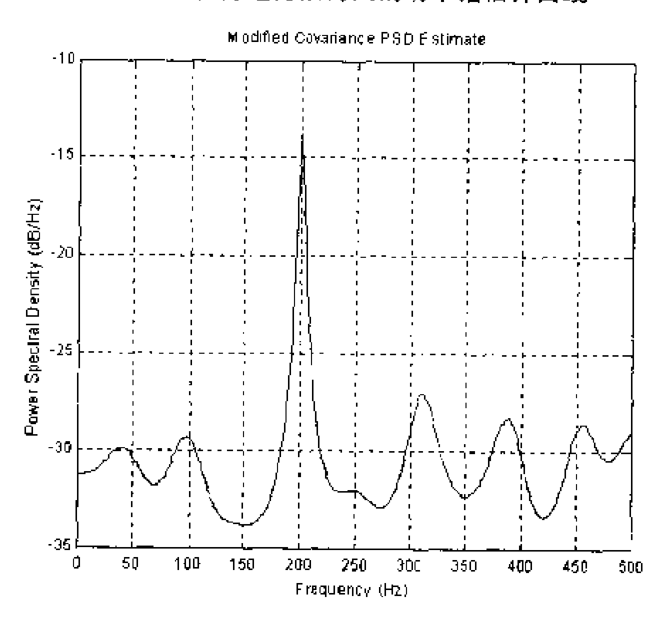


图8.13　改进协方差方法得到的功率谱估计曲线

8.3.6　AR 模型谱估计的性质

1.　AR谱的平滑特性

由于 AR 模型是一个有理分式，因而估计出的谱要比经典法的谱平滑。

例如，一含有噪声的余弦序列，分别采用周期图法与改进协方差法估计序列的功率谱，可以通过下面的程序实现：

```
%figure 8.14-15
Fs=1000;%采样频率
%产生序列
```

```
n=0:1/Fs:1;
xn=cos(2*pi*100*n)+randn(size(n));
%周期图法
figure(1)
%周期图法的参数设置
window=boxcar(length(xn));%窗函数
nfft=1024;%FFT 的点数
%周期图法计算功率谱
[Pxx,f]=periodogram(xn,window,nfft,Fs);
%绘制并标注图形
plot(f,10*log10(Pxx)),grid
xlabel('Frequency(Hz)');
ylabel('Power Spectral Density(dB/Hz)');
title('Periodogram PSD Estimate');
%改进协方差法
figure(2)
%改进协方差法的参数设置
order=20;%AR 模型的阶数
range='half';
magunits='db';
%改进协方差法估计并绘制功率谱
pmcov(xn,order,nfft,Fs,range,magunits)
```

得到的功率谱估计曲线如图 8.14 与图 8.15 所示。从图中明显可以看出，采用 AR 模型法得到的功率谱曲线要比周期图法得到的谱曲线平滑得多。

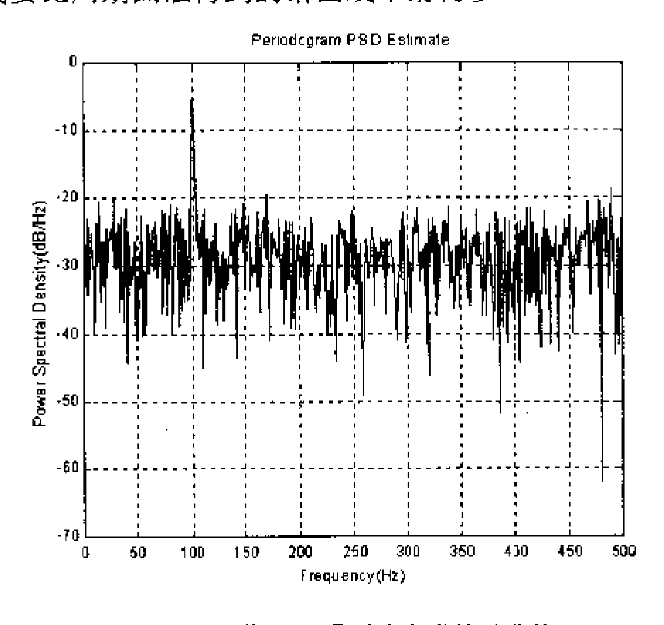


图8.14 周期图法得到功率谱估计曲线

2. AR谱的分辨率

经典谱估计的分辨率反比于有效信号的长度，但现代谱估计的分辨率可以不受此限制。这是因为，对给定的N点有限长序列$x(n)$，虽然其估计出的自相关函数也是有限长的，但现代谱估计的一些方法隐含着数据和自相关函数的外推，使其可能的长度超过给定的长度，因而 AR 谱的分辨率较高。

例如，设序列 $x(n)$ 由两个正弦信号组成，其频率分别为 $f_1=20\,\text{Hz}$，$f_2=21\text{Hz}$，并含有一定的噪声分量，试分别采用周期图法、Burg 法与改进协方差法估计序列的功率谱，且 AR 模型的阶数取 30 与 50 两种情况进行讨论。

上面的例子可以通过如下程序实现，得到的图形如图 8.16 至图 8.20 所示。

```
%figure 8.16-20
Fs=200;
n=0:1/Fs:1;
xn=sin(2*pi*20*n)+sin(2*pi*21*n)+0.1*randn(size(n));
window=boxcar(length(xn));
nfft=512;
[Pxx,f]=periodogram(xn,window,nfft,Fs);
figure(1)
plot(f,10*log10(Pxx)),grid
xlabel('Frequency(Hz)');
ylabel('Power Spectral Density(dB/Hz)');
title('Periodogram PSD Estimate');
order1=30;
order2=50;
range='half';
magunits='db';
figure(2)
pburg(xn,order1,nfft,Fs,range,magunits)
figure(3)
pburg(xn,order2,nfft,Fs,range,magunits)
figure(4)
pmcov(xn,order1,nfft,Fs,range,magunits)
figure(5)
pmcov(xn,order2,nfft,Fs,range,magunits)
```

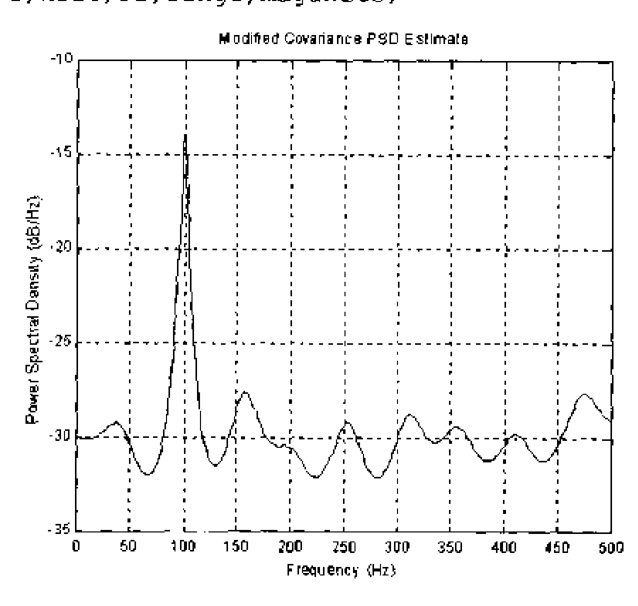


图8.15　改进协方差法功率谱估计曲线

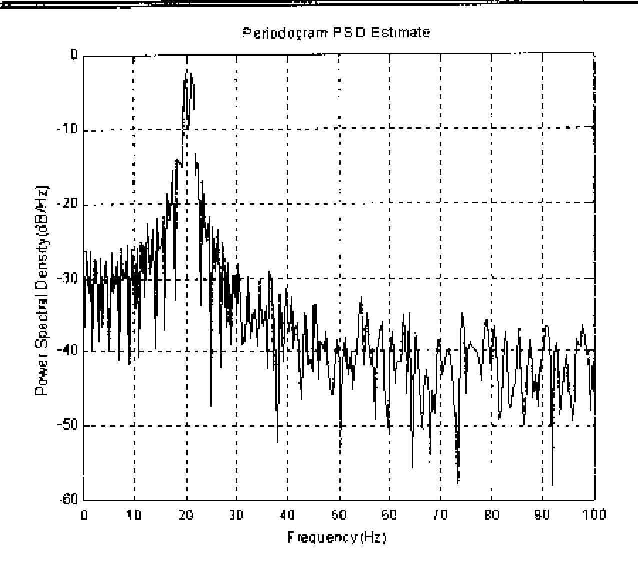


图8.16 周期图法求出的功率谱曲线

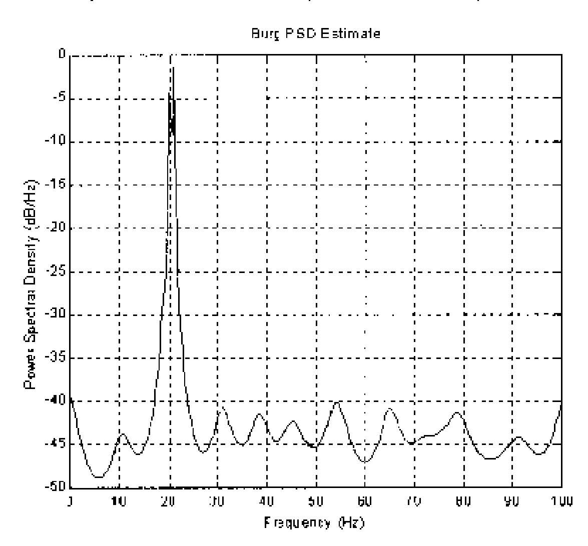


图8.17 Burg算法求出的AR功率谱曲线($P=30$)

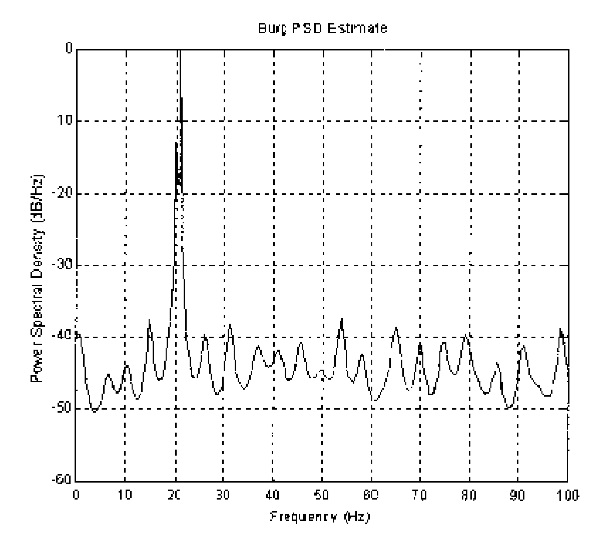


图8.18 Burg算法求出的AR功率谱曲线($P=50$)

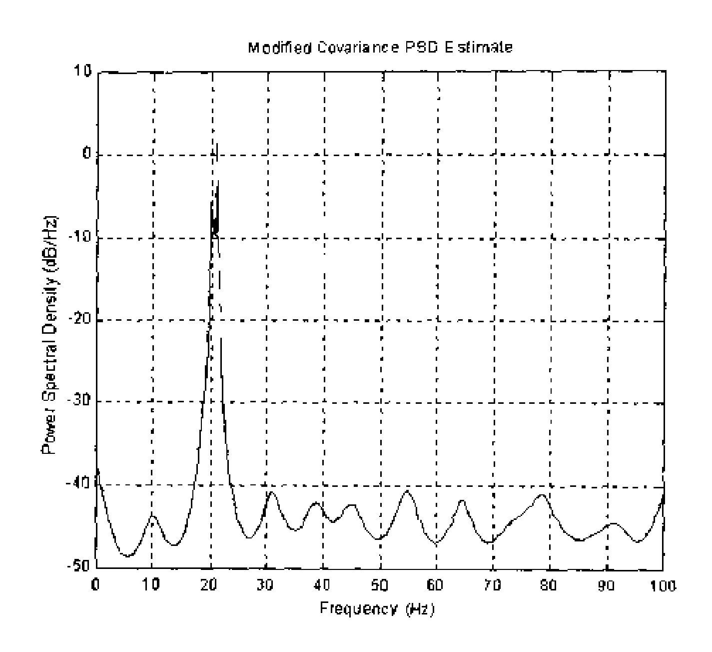


图8.19　改进协方差法求出的AR功率谱曲线（P = 30）

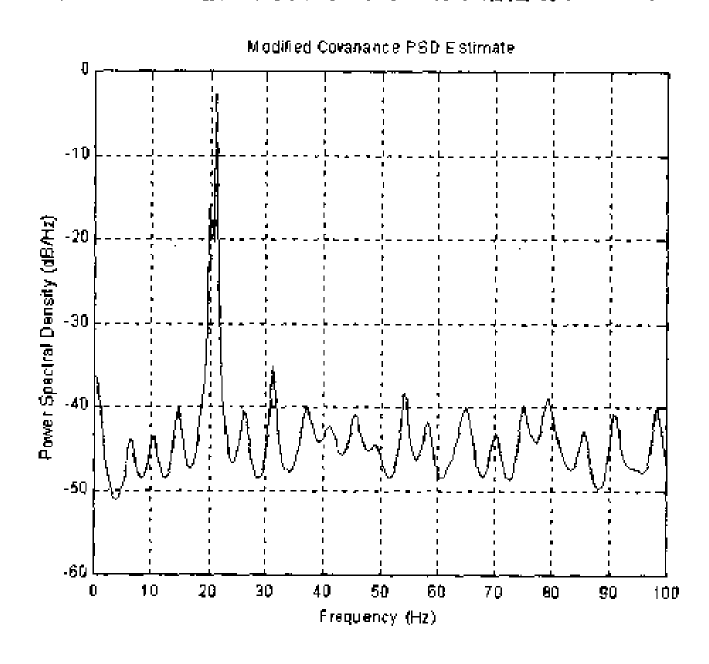


图8.20　改进协方差法求出的AR功率谱曲线（P = 50）

8.4　基于矩阵特征分解的功率谱估计

现代谱估计除了上一节讲的参数模型谱估计外，还有基于矩阵特征分解的功率谱估计，包括特征向量估计与 MUSIC 估计，这两种估计方法均为非参数估计方法，特征向量估计主要适用混有白噪声的正弦信号的功率谱估计，而 MUSIC 估计适合更为普遍情况下正弦信号参数估计的方法，它是多信号分类法的简称。

8.4.1 相关阵的特征分解

设序列$x(n)$是由M个复正弦加白噪声组成，那么其自相关函数为：

$$R_x(k)=\sum_{i=1}^{M}A_i\exp(jw_ik)+\sigma^2\delta(k) \tag{8.4.1}$$

式中A_i,w_i分别是第i个复正弦的功率及频率，σ^2是白噪声的方差。如果有$(p+1)$个$R_x(k)$组成相关阵：

$$R_{p+1}=\begin{bmatrix} R_x(0) & R_x^*(1) & \cdots & R_x^*(p) \\ R_x(1) & R_x(0) & \cdots & R_x^*(p-1) \\ \vdots & \vdots & & \vdots \\ R_x(p) & R_x(p-1) & \cdots & R_x(0) \end{bmatrix} \tag{8.4.2}$$

定义信号向量：

$$e_i=[1,\exp(jw_i),\cdots,\exp(jw_ip)]^{\mathrm{T}} \quad i=1,2,\cdots,M \tag{8.4.3}$$

则 $$R_{p+1}=\sum_{i=1}^{M}A_ie_ie_i^{\mathrm{H}}+\sigma^2I_{p+1} \tag{8.4.4}$$

令 $$S_{p+1}=\sum_{i=1}^{M}A_ie_ie_i^{\mathrm{H}} \tag{8.4.5}$$

将S_{p+1}作特征分解，得：

$$S_{p+1}=\sum_{i=1}^{p+1}\lambda_iV_iV_i^{\mathrm{H}} \tag{8.4.6}$$

V_i是对应于特征值λ_i的特征向量，且它们之间是相互正交的，即：

$$V_i^{\mathrm{H}}V_i=\begin{cases}1 & i=j \\ 0 & i\neq j\end{cases} \tag{8.4.7}$$

单位阵I_{p+1}也可用特征向量V_i表示为：

$$I_{p+1}=\sum_{i=1}^{p+1}V_iV_i^H \tag{8.4.8}$$

可以证明，S_{p+1}的秩最大为M，若M小于$p+1$，那么S_{p+1}将有$(p+1-M)$个零特征值，若将特征值按大小次序排列，那么S_{p+1}的特征分解可写为：

$$S_{p+1}=\sum_{i=1}^{M}\lambda_iV_iV_i^{\mathrm{H}} \tag{8.4.9}$$

$V_1,V_2,\cdots,V_M$称为主特征向量。

由上面的分析，得到：

$$R_{p+1}=\sum_{i=1}^{M}(\lambda_i+\sigma^2)V_iV_i^{\mathrm{H}}+\sum_{i=M+1}^{p+1}\sigma^2V_iV_i^{\mathrm{H}} \tag{8.4.10}$$

此式即为相关阵的特征分解。显然，R_{p+1}和信号矩阵S_{p+1}有着相同的特征向量。它们的所有特征向量$V_1,V_2,\cdots,V_{p+1}$形成了一个$p+1$维的向量空间，且它们是互相正交的。进一

步，该向量空间又可分成两个子空间，一个是由特征向量 $V_{M+1}, V_{M+2}, \cdots, V_{p+1}$ 张成的噪声空间，每个向量的特征值都是 σ^2；另一个是由主特征向量 $V_1, V_2, \cdots, V_M$ 张成的信号空间，其特征值分别是 $(\sigma^2+\lambda_1), (\sigma^2+\lambda_2), \cdots, (\sigma^2+\lambda_M)$，$\sigma^2$ 在此反映了噪声对信号空间的影响。

如果我们舍弃特征向量 $V_{M+1}, V_{M+2}, \cdots, V_{p+1}$，仅保留信号空间，可以得到相关阵的估计公式：

$$\hat{R}_{p+1} = \sum_{i=1}^{M}(\lambda_i + \sigma^2)V_i V_i^{\mathrm{H}} \tag{8.4.11}$$

基于矩阵 $\hat{R}_{p+1}$，再用以前所讲的任何一种方法估计 $x(n)$ 的功率谱，将得到好的谱估计。

8.4.2 MUSIC 谱估计方法

由于信号向量 e_i 和噪声空间的各个向量 $V_{M+1}, V_{M+2}, \cdots, V_{p+1}$ 都是正交的，因此，和它们的线性组合也是正交的，即：

$$e_i^{\mathrm{H}}\left(\sum_{k=M+1}^{p+1} a_k V_k\right) = 0 \qquad i = 1, 2, \cdots, M \tag{8.4.12}$$

令 $e(w) = [1, \exp(jw), \cdots, \exp(jwp)]^{\mathrm{T}}$，

则有：

$$e^{\mathrm{H}}(w)\left[\sum_{k=M+1}^{p+1} a_k V_k V_k^{\mathrm{H}}\right]e(w) = \sum_{k=M+1}^{p+1} a_k \left|e^{\mathrm{H}}(w)V_k\right|^2 \tag{8.4.13}$$

当 $w = w_i$ 时应为零，那么：

$$\hat{P}_x(w) = \frac{1}{\sum_{k=M+1}^{p+1} a_k \left|e^{\mathrm{H}}(w)V_k\right|^2} \tag{8.4.14}$$

在 $w = w_i$ 处，应是无穷大，但由于 V_k 是由相关阵分解得到的，而相关阵是估计出的，因此必有误差，所以 $\hat{P}_x(w_i)$ 为有限值，但呈现尖的峰值，其峰值对应的频率即是正弦信号的频率，由此也可得到序列 $x(n)$ 的功率谱估计。其功率谱的分辨率要好于 AR 模型。

(1) 若令 $a_k = 1, k = M+1, \cdots, p+1$，所得估计即为 MUSIC 估计，即：

$$\hat{P}_{\mathrm{MUSIC}}(w) = \frac{1}{e^{\mathrm{H}}(w)\left(\sum_{k=M+1}^{p+1} V_k V_k^{\mathrm{H}}\right)e(w)} \tag{8.4.15}$$

(2) 若令 $a_k = 1/\lambda_k, k = M+1, \cdots, p+1$，则所得功率谱称特征向量估计，即：

$$\hat{P}_{\mathrm{EV}}(w) = \frac{1}{e^{\mathrm{H}}(w)\left(\sum_{k=M+1}^{p+1} \frac{1}{\lambda_k} V_k V_k^{\mathrm{H}}\right)e(w)} \tag{8.4.16}$$

8.4.3 MUSIC 估计与特征向量估计的 MATLAB 实现

MATLAB 中有两个函数用来实现基于矩阵特征分解的功率谱估计：函数 Pmusic 与函

数 Peig，函数 Pmusic 为 MUSIC 估计，而函数 Peig 为特征向量估计。

1. Pmusic函数

功能：利用 MUSIC 法进行功率谱估计。

格式：

```
Pxx = Pmusic (x,P,NFFT)
Pxx = Pmusic (x,[P THRESH],NFFT)
[Pxx,W] = Pmusic (x,P,NFFT)
[Pxx,F] = Pmusic (x,P,NFFT,Fs)
[Pxx,F] = Pmusic (x,P,NFFT,Fs,NW,NOVERLAP)
[Pxx,F,V,E] = Pmusic (...)
Pmusic (x,P,NFFT,Fs,NW,NOVERLAP)
```

说明：

Pxx = Pmusic (x,P,NFFT)采用 MUSIC 法估计向量 x 的功率谱，若 x 为实信号，进行单边功率谱估计，若 x 为复信号，进行双边功率谱估计；参数 P 用来指定信号空间中特征向量的数目；NFFT 为 FFT 算法的长度，其默认值为 256，若 NFFT 为偶数，则 Pxx 为 NFFT/2+1 维的列矢量，若 NFFT 为奇数，则 Pxx 为(NFFT+1)/2 维的列矢量；当 x 为复数时，Pxx 的长度为 NFFT。

Pxx = Pmusic (x,[P THRESH],NFFT)中，用所有大于参数 THRESH 与最小特征向量之积的特征向量作为主特征向量，则信号空间的最大维数为 P。

[Pxx,W] = Pmusic (x,P,NFFT)中，返回一个频率向量 W，若 x 为实信号，在区间[0 pi]上进行功率谱估计，若 x 为复信号，则在区间[0 2pi]上进行功率谱估计。

[Pxx,F] = Pmusic (x,P,NFFT,Fs)中，可在 F 向量得到功率谱估计的频率点，Fs 指定采样频率，若 x 为实信号，在区间[0 Fs/2]上进行功率谱估计，若 x 为复信号，则在区间[0 Fs]上进行功率谱估计。

[Pxx,F] = Pmusic (x,P,NFFT,Fs,NW,NOVERLAP)中，将向量 x 分成长度为 NW 的各段，每一段之间用 NOVERLAP 个部分重叠，然后以各段为列向量组成矩阵，最后进行功率谱估计，参数 NW 的默认值为 2*P，参数 NOVERLAP 的默认值为 NW-1。

[Pxx,F,V,E] = Pmusic (...)中，返回特征向量 V 与特征值 E。

Pmusic (x,P,NFFT,Fs,NW,NOVERLAP)中，没有输出参数，则直接给出功率谱估计曲线。

2. Peig函数

功能：进行特征向量谱估计。

格式：

```
Pxx = Peig(x,P,NFFT)
Pxx = Peig (x,[P THRESH],NFFT)
[Pxx,W] = Peig (x,P,NFFT)
[Pxx,F] = Peig (x,P,NFFT,Fs)
[Pxx,F] = Peig (x,P,NFFT,Fs,NW,NOVERLAP)
[Pxx,F,V,E] = Peig (...)
Peig (x,P,NFFT,Fs,NW,NOVERLAP)
```

说明：

Peig 函数与 Pmusic 函数只是采用的方法不同，其格式与参数说明均相同。

例如，设序列 $x(n)$ 由两个正弦信号组成，其频率分别为 f_1=200Hz，f_2=205Hz，采样频率为 Fs=1000Hz，并含有一定的噪声分量，试通过此例说明 Peig 函数与 Pmusic 函数的用法，并比较基于矩阵特征分解谱估计的谱分辨率与 AR 模型中 Yule-Walker 法谱估计的谱分辨率。

可以通过下面的 MATLAB 程序实现：

```
%figure 8.21-23
%采样频率
Fs=1000;
%产生序列
n=0:1/Fs:1;
xn=sin(2*pi*200*n)+sin(2*pi*205*n)+0.1*randn(size(n));
%Yule-Walker 法进行谱估计
figure(1)
%第一步：设置参数
order=30;
nfft=1024;
range='half';
magunits='db';
%第二步：利用 Pyulear 函数估计并绘制功率谱曲线
pyulear(xn,order,nfft,Fs,range,magunits)
%基于矩阵特征分解的谱估计
%第一步：设置参数
p=25;%主特征向量的个数
%第二步：MUSIC 估计
figure(2)
pmusic(xn,p,nfft,Fs)
%第三步：特征向量估计
figure(3)
peig(xn,p,nfft,Fs)
```

得到的功率谱估计曲线如图 8.21～8.23 所示。从图中可以看出，基于矩阵特征分解谱估计的谱分辨率要好于 AR 模型中 Yule-Walker 法谱估计的谱分辨率。

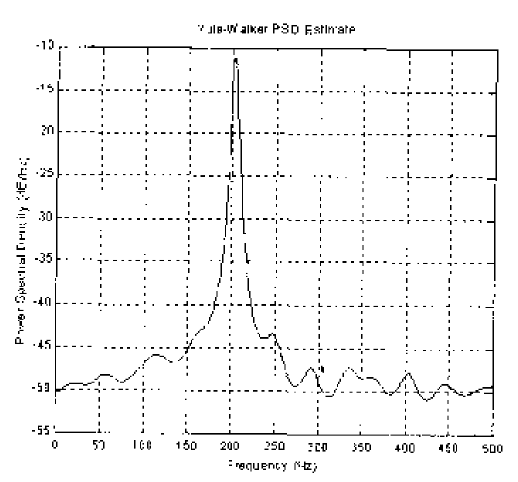


图8.21　Yule-Walker法得到的功率谱曲线

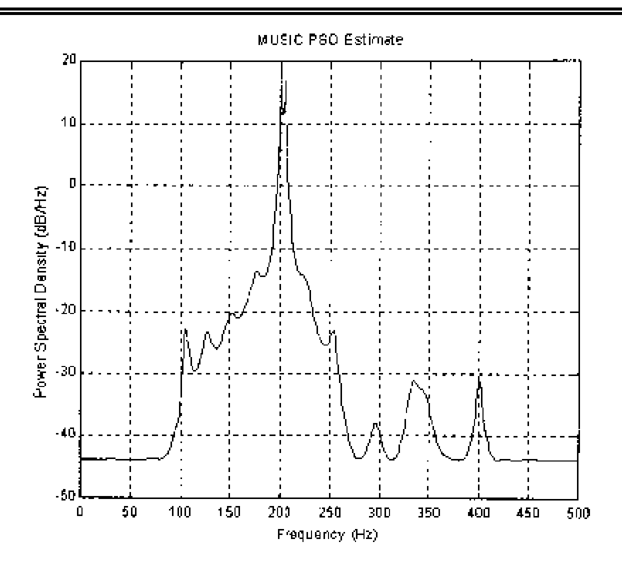


图8.22 MUSIC法得到的功率谱曲线

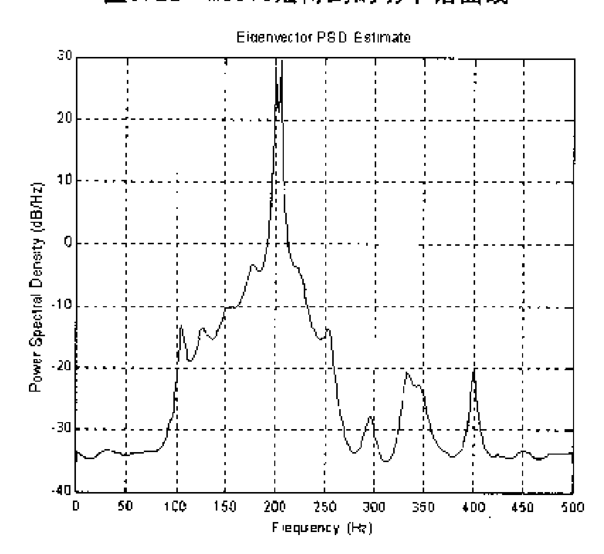


图8.23 特征向量估计法得到的功率谱曲线

附录 A　MATLAB 6 命令参考

MATLAB 6 提供了 24 类基本命令函数，它们有一部分是 MATLAB 的内部命令，有一部分是以 M 文件形式出现的函数，这些 M 文件按不同类别归于一子目录下，每个目录中除了以 M 文件表示的函数命令之外，还有一个特殊的文件 Contens.m，它给出了该目录中所有 M 文件的简单注释，用来说明 M 文件的用途。每一个 M 文件中都有一部分用来说明该函数的格式、参数的含义以及相关函数的名称等，可以通过这个命令：

```
help  functionname
```

来查看有关函数 functionname 的帮助信息，functionname 为 M 文件名。

限于篇幅，本附录不给出各个函数的详细说明，读者可利用 help 命令获得这些信息。

表 A.1 为 MATLAB 6 中 24 类基本命令函数的子目录及其含义，表 A.2～A.24 中列出各函数的简要说明，以供读者参考。

表 A.1　基本命令函数目录

目录名称	命令函数	索　引
audio	音频处理函数	表 A.2
datafun	数据分析与傅立叶变换函数	表 A.3
datatypes	数据类型与结构	表 A.4
elfun	基本数学函数	表 A.5
elmat	基本矩阵与矩阵操作	表 A.6
funfun	功能函数与微分方程的求解	表 A.7
general	通用命令	表 A.8
graph2d	二维图形函数	表 A.9
graph3d	三维图形函数	表 A.10
graphics	通用图形函数	表 A.11
iofun	文件 I/O 函数	表 A.12
Lang	语言结构与调试	表 A.13
matfun	矩阵函数-数值线性代数	表 A.14
Ops	操作符与特殊字符	表 A.15
polyfun	多项式与内插函数	表 A.16
sparfun	稀疏矩阵函数	表 A.17
specfun	特殊数学函数	表 A.18
specgraph	特殊图形函数	表 A.19
strfun	字符串函数	表 A.20
timefun	时间与日期函数	表 A.21

续表

目录名称	命令函数	索　引
Uitools	图形用户界面工具	表 A.22
verctrl	版本控制	表 A.23
winfun	Windows 操作系统文件界面	表 A.24
demos	演示实例	——

表 A. 2　音频处理函数

● 音频硬件驱动器函数		
	saxis	音频轴刻度
	sound	变向量为音频信号
	soundsc	自动变向量为音频信号
	waveplay	用 Windows 音频输出装置播放音频信号
	waverecord	用 Windows 音频输入装置播放录音
● 音频文件的输入输出函数		
	auread	读按 Wu-law 编码的音频格式
	auwrite	写按 Wu-law 编码的音频格式
	wavread	读 MS Windows 的.WAV 音频信号
	wavwrite	写 MS Windows 的.WAV 音频信号
● 变换函数		
	lin2mu	变线性音频信号为 Wu-law 编码音频信号
	mu2lin	变 Wu-law 编码音频信号为线性音频信号

表A. 3　数据分析与傅立叶变换函数

● 基本操作		
	conv2	求元素累积积
	max	求最大值
	min	求最小值
	mean	求均值
	median	求中值
	std	求标准差
	var	求方差
	Sort	按升序排列
	sortrow	以行按升序排列矩阵
	sum	求各元素之和
	prod	求各元素之积
	cumsum	求元素累积和
	cumprod	求元素累积积

续表

● 基本操作		
	trapz	利用梯形法计算数值积分
	cumtrapz	利用梯形法计算数值累积积分
● 有限差分		
	Diff	计算差分和近似微分
	gradient	计算近似梯度
	del2	离散拉普拉斯变换
● 相关		
	corroef	求相关系数
	cov	求协方差矩阵
	subpace	子空间之间的夹角
● 滤波和卷积		
	filter	求各元素之和
	filter2	求各元素之积
	conv	求元素累积和
	conv2	求元素累积积
	deconv	利用梯形法计算数值积分
● 傅立叶变换		
	fft	离散傅立叶变换
	fft2	二维离散傅立叶变换
	ifft	离散逆傅立叶变换
	ifft2	二维离散逆傅立叶变换
	fftshift	将直流分量平移到频谱中心

表 A.4 数据类型与结构

● 数据类型		
	double	双精度形式
	sparse	产生稀疏矩阵
	char	产生字符数组
	cell	产生单元数组
	struct	产生或转换成结构体
	single	单精度形式
	uint8	无符号的 8 位整数形式
	uint16	无符号的 16 位整数形式
	uint32	无符号的 32 位整数形式
	int8	有符号的 8 位整数形式
	int16	有符号的 16 位整数形式

续表

● 数据类型		
	int32	有符号的 32 位整数形式
	inline	构造 INLINE 对象
	function_handle	句柄函数
	JaveArray	产生 Java 数组
	javaMethod	调用 Java 方法
	javaObject	调用 Java 对象构造器
● 多维数组函数		
	cat	数组连接
	ndims	数组的维数
	ndgrid	生成 N 元函数及其插值用的数据
	permute	排列数组维数
	ipermute	逆排列数组维数
	shiftdim	改变数组维数
	squeeze	删除单一维数
● 单元数组函数		
	cell	产生单元数组
	cellfun	用单元函数操作单元数组
	celldisp	显示单元数组的内容
	cellplot	用图形方式显示单元数组的结构
	num2cell	将数值型数组转换成单元数组
	deal	处理输入数据产生新输出数据
	cell2struct	将单元数组转换成结构体
	struct2cell	将结构体转换成单元数组
	iscell	若是单元数组返回 TURE
● 结构体函数		
	fieldnames	获取结构体的字段名
	getfield	获取结构体的字段内容
	setfield	设置结构体的内容
	rmfield	删除结构体中的字段
	isfield	若是结构体中的字段返回 TRUE
	isstruct	若是结构体返回 TRUE
● 句柄函数		
	@	产生函数句柄
	Func2str	将函数句柄数组转换成字符
	Str2func	将字符转换成函数句柄数组
	Functions	列出与句柄相关的函数

续表

● 重载操作		
	plus	两个运算元的加法
	minus	两个运算元的减法
	uminus	单个运算元的相反数
	uplus	单个运算元前加正号
	times	两个矢量中对应元素相乘
	mtimes	矩阵乘法
	rdivide	两个矢量中对应元素右乘法
	ldivide	两个矢量中对应元素左乘法
	mrdivide	矩阵右乘法
	mldivide	矩阵左乘法
	power	单个元素的乘方
	mpower	矩阵的乘方
	lt	小于
	gt	大于
	le	小于等于
	ge	大于等于
	ne	不等于
	eq	等于
	and	逻辑与运算
	or	逻辑或运算
	not	逻辑非运算
● 对象导向设计函数		
	class	产生对象或返回对象类
	methods	列出类名及其特性
	methodsview	查看类名及其特性
	isa	若对象属于给定类返回 TRUE
	isjava	若是 Java 对象返回 TRUE
	isobject	若是 MATLAB 对象返回 TRUE
	inferiorto	低级类关系
	superiorto	高级类关系

表 A.5 基本数学函数

● 三角函数		
	sin	正弦
	sinh	双曲正弦

续表

● 三角函数		
	asin	反正弦
	asinh	反双曲正弦
	cos	余弦
	cosh	双曲余弦
	acos	反余弦
	acosh	反双曲余弦
	tan	正切
	tanh	双曲正切
	atan	反正切
	atan2	四象限反正切
	atanh	反双曲正切
	sec	正割
	sech	双曲正割
	asec	反正割
	asech	反双曲正割
	csc	余割
	csch	双曲余割
	acsc	反余割
	acsch	反双曲余割
	cot	余切
	coth	双曲余切
	acot	反余切
	acoth	反双曲余切
● 指数函数		
	exp	指数
	log	自然对数
	log10	常用对数
	log2	以 2 为底的对数或分割浮点数
	pow2	2 的次幂或比例浮点数
	sqrt	平方根
● 复数函数		
	abs	绝对值
	angle	相角
	complex	由实虚部构造复数

续表

● 复数函数		
	Conj	复共轭
	imag	复数虚部
	real	复数实部
	isreal	若是实数组返回 TRUE
	cplxpair	将数值分成复共轭对
● 取整与取余函数		
	fix	朝零方向取整
	floor	朝负无穷大方向取整
	ceil	朝正无穷大方向取整
	round	朝最近的整数取整
	mod	取余
	rem	除后取余

表 A.6 基本矩阵与矩阵操作

● 基本矩阵		
	zeros	零矩阵
	ones	全“1”矩阵
	eye	单位矩阵
	repmat	复制矩阵
	rand	均匀分布的随机数矩阵
	randn	正态分布的随机数矩阵
	linspace	线性间隔的向量
	logspace	对数间隔的向量
	freqspace	产生频率间隔向量
	meshgrid	三维图形的 X 和 Y 数组
	:	规则间隔的向量
● 基本数组信息		
	size	矩阵的大小
	length	向量的长度
	ndims	矩阵的维数
	disp	显示文本矩阵
	isempty	若矩阵为空返回 TRUE
	isequal	若数组相同返回 TRUE
	isnumeric	若数组的元素为数值返回 TRUE
	islogical	若数组合乎逻辑返回 TRUE
	logical	将数组转换成逻辑数组

续表

● 矩阵操作	
reshape	改变矩阵大小
diag	产生或提取对角阵
blkdiag	分块对角阵串联
fliplr	矩阵作左右翻转
flipud	矩阵作上下翻转
flipdim	沿指定维数翻转矩阵
rot90	矩阵旋转 90°
tril	提取矩阵的下三角部分
triu	提取矩阵的上三角部分
:	矩阵的索引号
find	查找元素的索引号
end	最后的索引号
sub2ind	多个下标的线性索引
ind2sub	线性索引的多个下标
● 特殊变量与常数	
ans	当前的结果
eps	相对浮点数
realmax	最大浮点数
realmin	最小浮点数
pi	圆周率
i , j	虚数单位
inf	无穷大
NaN	非数值
isnan	若为非数值返回 TRUE
isinf	若含有无穷多个元素返回 TRUE
isfinite	若含有有限个元素返回 TRUE
flops	浮点运算次数
why	简明的答案
● 特殊矩阵	
company	友矩阵
gallery	几个小的测试矩阵
hadamard	Hadamard 矩阵
hankel	Hankel 矩阵

续表

● 特殊矩阵		
	hilb	Hilbert 矩阵
	invhilb	逆 Hilbert 矩阵
	Kron	kronecher 张量积
	Magic	魔方矩阵
	Pascal	Pascal 矩阵
	Rosser	经典的对称特征值测试问题
	toeplitz	Toeplitz 矩阵
	vander	Vandermonde 矩阵
	wilkinson	Wilkinson 特征值测试矩阵

表 A.7　功能函数与微分方程的求解

● 功能函数		
	fminbnd	一元函数的极小化
	fminsearch	多元函数的极小化
	Fzero	求一元函数的零点
● 属性设置函数		
	optimset	产生或改变优化属性结构
	optimget	从属性结构中获取优化参数
	odeset	产生或改变常微分方程的属性结构
	Odeget	获取常微分方程的属性结构
	bvpset	产生或改变边界值问题的属性结构
	bvpget	获取边界值问题的属性结构
● 内联函数对象		
	inline	构造内联函数对象
	argnames	变量名
	formula	函数格式
	Char	将内联对象转换成字符数组
● 数值积分函数		
	quad	低阶法计数值积分
	quadl	高阶法计数值积分
	Dblquad	计算二重积分
● 微分函数		
	ode45	中阶法求解非 Stiff 微分方程
	ode23	低阶法求解非 Stiff 微分方程
	ode113	变阶法求解非 Stiff 微分方程

续表

● 微分函数		
	ode23t	适度求解 Stiff 微分方程
	ode15s	变阶法求解 Stiff 微分方程
	ode23s	低阶法求解 Stiff 微分方程
	ode23tb	低阶法求解 Stiff 微分方程
● 输入输出函数		
	odefile	ODE 文件模板
	odeplot	ODE 输出函数的时间轨迹图
	odephas2	ODE 输出函数的二维相平面图
	odephas3	ODE 输出函数的三维相平面图
	odeprint	在 MATLAB 指令窗显示结果
● 辅助函数		
	fcnchk	检查 FUNFUN 中的函数
	symvar	列出符号变量
	isvarname	若为有效变量名返回 TRUE
	vectorize	串表达式或内联函数适于数组运算
	inlineeval	判断超出内联路径的内联函数
● 绘图函数		
	ezplot	画二维曲线的简捷指令
	ezplot3	画三维曲线的简捷指令
	ezpolar	画极坐标图的简捷指令
	ezcontour	画等位线的简捷指令
	ezcontourf	画填色等位线的简捷指令
	ezmesh	画网线图的简捷指令
	ezmeshc	画带等位线的网线图的简捷指令
	ezsurf	画表面图的简捷指令
	ezsurfc	画带等位线的表面图的简捷指令
	fplot	返函绘图指令

表 A.8 通用命令

● 一般信息		
	Help	在线帮助命令
	helpwin	交互式在线帮助命令
	helpdesk	打开超文本形式用户指南
	support	打开 MathWorks 技术支持网页

续表

● 一般信息		
	Demo	运行演示示例
	Java	在 MATLAB 中使用 Java
	Ver	版本信息
	whatsnew	在说明书中未包含的新信息
● 管理变量与工作空间		
	who	列出当前变量
	whos	详细列出当前变量
	Workspace	启动工作空间
	Clear	从内存中清除变量与函数
	Pack	整理工作空间内存
	Load	从磁盘文件中恢复变量
	Save	保存工作空间变量
	Quit	退出 MATLAB
● 管理命令		
	what	M、MAT、EXE 文件的目录列表
	type	列出 M 文件
	Edit	启动 M 文件编辑器
	open	打开 M 文件
	which	定位函数与文件
	pcode	创建预解译 P 码文件
	inmem	列出内存中的函数名
	Mex	编译 MEX 函数
● 管理搜索路径函数		
	path	控制 MATLAB 的搜索路径
	addpath	增加路径到搜索路径
	rmpath	从搜索路径中删除路径
	pathtool	搜索路径管理器
	rehash	刷新函数及文件系统缓冲器
	import	为当前范围输入 Java 包
● 与文件和操作系统有关的命令		
	Cd	改变当前工作目录
	copyfile	复制文件
	pwd	显示当前工作目录
	Dir	目录列表
	delete	删除文件

续表

● 与文件和操作系统有关的命令		
	Getenv	获取环境变量值
	Mkdir	创建目录
	!	执行操作系统命令
	dos	执行 DOS 命令并返回结果
	unix	执行 UNIX 命令并返回结果
	vms	执行 VMS DCL 命令并返回结果
	web	打开 WEB 浏览器
	computer	计算机的类型
	isunix	若是 MATLAB 的 UNIX 版返回 TRUE
	ispc	若是 MATLAB 的 PC 版返回 TRUE
	isstudent	若是 MATLAB 学生版返回 TRUE
● 控制命令窗口		
	echo	MATLAB 文件内使用的回显命令
	more	在命令窗口中控制分页输出
	Diary	保存 MATLAB 任务
	format	设置输出格式
	Beep	产生嘟嘟声
● 调试 M 文件		
	debug	列出调试命令
	dbstop	设置断点
	dbclear	删除断点
	dbcont	继续执行
	dbdown	改变局部空间的内容
	dbstack	列出调用者
	dbstatus	列出所有断点
	Dbstop	执行一条或多条语句
	dbtype	带行号列出 M 文件
	Dbup	改变局部空间的内容
	dbquit	退出调试状态
	dbmex	调试 MEX 文件
● 定位 M 文件的依赖函数		
	depfun	定位 M 文件的依赖函数
	Depdir	定位 M 文件的依赖目录

续表

● 其他函数		
	binpatch	修补二进制文件
	doc	装载 HTML 文档
	Docroot	决定 MATLAB 帮助根目录
	Exit	退出 MATLAB
	helpinfo	关于帮助的信息
	helpview	在帮助浏览器中显示 HTML 文件
	info	关于 MATLAB 与 MathWorks 的信息
	isvms	若是 MATLABVMS 版返回 TRUE
	isppc	若是 Macintosh PowerPC 返回 TRUE
	isieee	若计算机执行 IEEE 规则返回 TRUE
	lookfor	按关键字搜索 M 文件
	ls	列出目录
	MATLABpath	MATLAB 的搜索路径
	Memory	内存限制帮助
	notebook	启动 MATLAB 和 Word 的集成环境
	Nnload	装载 Netscape 浏览器
	prepender	效用函数
	Saveas	按期望的输出格式保存图形或模型
	subscribe	订阅 MathWorks 的时事通讯

表 A.9　二维图形函数

● 基本 X—Y 函数		
	plot	线性图形
	loglog	对数坐标图形
	semilogx	X 轴为对数坐标的图形
	semilogy	Y 轴为对数坐标的图形
	polar	极坐标图形
	plotyy	双纵坐标图形
● 坐标轴与图形控制		
	axis	控制坐标轴的范围与外观
	zoom	图形的放大和缩小
	grid	画分格线
	box	框状坐标轴
	hold	保留现有图形
	axes	创建轴对象的低层指令
	subplot	画子图

续表

● 图形注释

	plotedit	图形注释与编辑工具
	legend	图形图例
	title	给图形添加标题
	xlabel	标注 X 坐标轴
	ylabel	标注 Y 坐标轴
	texlabel	从字符串中产生 TeX 格式
	text	文本注释
	gtext	用鼠标放置文本

表 A.10 三维图形函数

● 基本三维图形

	plot3	在三维空间中绘制曲线与点
	mesh	三维网格曲面
	surf	三维曲面阴影图
	fill3	填充三维多边形

● 颜色控制

	colormap	颜色对照表
	caxis	伪彩色坐标轴刻度
	shading	彩色阴影方式
	hidden	设置网格消隐方式
	Brighten	使彩色板变亮或变暗
	Colordef	设置颜色的默认值
	graymon	为灰度监视器设置图形默认值

● 颜色板

	hsv	色彩与饱和度颜色板
	Hot	黑、红、黄与白颜色板
	gray	线性灰度颜色板
	bone	以黄色为基调的灰度颜色板
	copper	线性青铜色调的颜色板
	pink	线性粉红阴影颜色板
	white	全白色颜色板
	flag	红、白、蓝、黑交替的颜色板
	lines	采用 plot 画线色
	colorcube	三浓淡多彩交叉色颜色板
	Vga	16 色的 Windows 颜色板

续表

● 颜色板		
	Jet	蓝头红尾饱和色
	prism	光谱色颜色板
	cool	青紫调冷色图
	autumn	红黄调秋颜色板
	spring	紫黄调春颜色板
	winter	蓝绿调冬颜色板
	summer	绿黄调夏颜色板
● 透明度控制		
	Alpha	透明度模式
	alphamap	透明度查寻表
	Alim	透明度缩放比例
● 亮度控制		
	Surfl	带亮度的三维曲面阴影图
	lighting	加亮模式
	material	材料反射模式
	specular	镜面反射模式
	diffuse	漫反射模式
	surfnorm	曲面法线
	camlight	创建或设置灯光位置
	lightangle	灯光的球面位置
● 坐标轴控制		
	axis	控制坐标轴的范围与外观
	zoom	图形的放大和缩小
	grid	画分格线
	box	框状坐标轴
	hold	保留现有图形
	axes	创建轴对象的低层指令
	subplot	画子图
	daspect	数据纵横比
	pbaspect	画出框状纵横比
	xlim	X 的限制
	ylim	Y 的限制
	zlim	Z 的限制

续表

● 视点控制		
	view	指定三维图形视点
	viewmtx	显示变换矩阵
	rotate3d	旋转三维图形的视点
● 相机控制		
	campos	相机位置
	camtartet	相机目标
	camva	相机视角
● 图形注释		
	Plotedit	图形注释与编辑工具
	legend	图形图例
	title	给图形添加标题
	xlabel	标注 X 坐标轴
	ylabel	标注 Y 坐标轴
	texlabel	从字符串中产生 TeX 格式
	text	文本注释
	gtext	用鼠标放置文本

表 A. 11 通用图形函数

● 句柄图形操作		
	Set	设置对象属性
	Get	获取对象属性
	reset	重置对象属性
	delete	删除对象
	Gco	获取当前对象的句柄
	gcbo	获取当前调用对象的句柄
	gcbf	获取当前调用图形的句柄
	drawnow	填充未完成绘图事件
	findobj	寻找指定特性值的对象
	isappdata	检查指定应用数据是否存在
	getappdata	获取指定应用数据的值
	setappdata	设置指定应用数据
	rmappdata	删除指定应用数据
● 建立与控制图形窗口		
	figure	建立图形
	Gcf	获取当前图形的句柄

续表

● 建立与控制图形窗口		
	Clf	清除当前图形
	shg	显示图形窗口
	close	关闭图形
	refresh	刷新图形
	openfig	打开已存图形的备份
● 建立与控制坐标系		
	subplot	在标定位置上建立坐标系
	Axes	在任意位置上建立坐标系
	Gca	获取当前坐标系的句柄
	Cla	清除当前坐标系
	axis	控制坐标系的刻度与外观
	box	框状坐标轴
	caxis	控制伪彩色坐标刻度
	hold	保持当前图形
	ishold	返回保持状态
● 句柄图形对象		
	Line	建立曲线
	Text	建立文本串
	patch	建立图形填充块
	rectangle	建立矩形、圆或椭圆
	surface	建立曲面
	image	建立图像
	uicontrol	建立用户界面控制
	unimenu	建立用户界面菜单
	Uicontextmenu	建立用户上下文菜单
● 打印与存储		
	Print	打印或保存图形
	printopt	设置本地打印机的默认值
	Orient	设置纸张取向
● 多媒体用户函数		
	actxcontrol	创建多媒体控件
	actxserver	创建多媒体服务器

表 A.12 文件 I/O 函数

● 文件 I/O		
	dlmread	读定界限的文本文件

续表

● 文件 I/O		
	Dlmwrite	写定界限的文本文件
	Load	加载 MAT 文件到工作空间
	importdata	加载磁盘文件到工作空间
	wk1read	读 WK1 文件
	wk1write	写 WK1 文件
	xlsread	读 XLS 文件
● 图形文件 I/O		
	imfinfo	返回图形文件的信息
	imread	从图形文件中读图像
	imwrite	往图形文件中写图像
● 音频文件 I/O		
	auread	读.au 声音文件
	auwrite	写.au 声音文件
	wavread	读.wav 声音文件
	wavwrite	写.wav 声音文件
● 视频文件 I/O		
	aviread	读.avi 文件
	aviinfo	返回.avi 文件的信息
	avifile	创建新的.avi 文件
	movie2avi	在 MATLAB 中创建 AVI 动画
● 格式化文件 I/O		
	Fgetl	从文件中读行，并丢弃换行符
	Fgets	从文件中读行，并保持换行符
	fprintf	将格式化数据写入到文件
	fscanf	从文件中读格式化数据
	input	提示用户输入
	textread	从文本文件中读格式化数据
● 字符串变换		
	sprintf	将格式化数据写到字符串
	Sscanf	从格式化字符串中读取
	strread	从文本串中读格式化数据
● 打开与关闭文件		
	Fopen	打开文件
	fclose	关闭文件

续表

● 二进制文件 I/O		
	Fread	读二进制文件
	fwrite	写二进制文件
● 文件定位		
	Feof	测试文件尾
	Ferror	查询文件出错状态
	Frewind	反绕文件
	Fseek	设置文件位置指针
	Ftell	获取文件位置指针
● 文件名处理		
	fileparts	部分文件名
	Filesep	此平台的隔离目录
	Fullfile	从部分文件名中建立全名
	MATLABroot	MATLAB 的安装根目录
	Mexext	此平台的 MEX 文件扩展
	partialpath	局部路径名
	pathsep	此平台的隔离路径
	Prefdir	优先目录名
	tempdir	获取临时目录
	tempname	获取临时文件
● 命令窗口 I/O		
	Clc	清除命令窗口
	Disp	显示矢量
	Home	发送指针
	Input	提示用户输入
	Pause	等待用户

表 A. 13 语言结构与调试

● 程序流控制		
	If	条件执行语句
	else	与 if 命令配合使用
	elseif	与 if 命令配合使用
	end	for、while、if 语句的结束
	for	重复执行指定次数
	while	重复执行不定次数
	break	终止循环语句

续表

● 程序流控制		
	continue	继续执行
	switch	条件转换语句
	case	与 switch 命令配合使用
	otherwise	与 switch 命令配合使用
	try	开始 TRY 块
	catch	开始 CATCH 块
	return	返回引用函数
● 赋值与执行		
	eval	执行由 MATLAB 表达式构成的字串
	evalc	执行 MATLAB 表达式
	feval	执行由字串指定的函数
	evalin	执行工作空间中的表达式
	builtin	采用重载方法执行内在函数
	assignin	在工作空间中分配变量
	run	执行
● 参数处理		
	nargchk	检查输入变量数
	nargoutchk	检查输出变量数
	nargin	输入变量数
	nargout	输出变量数
	Varargin	变长度输入变量
	varargout	变长度输出变量
	inputname	输入变量名
● 显示消息		
	Error	出错消息
	warning	警告消息
	lasterr	最后一个错误消息
	lastwarn	最后一个警告消息
	Disp	显示消息
	display	显示消息的重载函数
	fprintf	显示格式化消息
	sprintf	将格式化数据写到字符串

续表

● 交互输入		
	Input	提示用户输入
	keyboard	使用键盘输入
	Pause	等待用户
	uimenu	建立用户界面菜单
	uicontrol	建立用户界面控制
● 下标、函数与变量		
	script	关于 MATLAB 手稿与 M 文件
	function	增加新的函数
	global	定义全局变量
	persistent	定义永久变量
	mfilename	当前执行的 M 文件名
	Lists	逗号隔离的列表
	Exist	检查变量或函数是否定义
	isglobal	若为全局变量返回 TRUE
	Mlock	防止 M 文件被清除
	munlock	允许 M 文件被清除
	mislocked	若 M 文件不可清除返回 TRUE
	precedence	MATLAB 中的优先权
	isvarname	检查是否为有效变量名
	iskeyword	检查是否为键盘输入

表 A.14 矩阵函数— 数值线性代数

● 矩阵分析		
	Norm	计算矩阵或向量范数
	normest	矩阵 2 范数估计
	rank	计算矩阵的秩
	det	计算矩阵的行列式
	trace	计算矩阵的迹
	null	零矩阵
	orth	正交化
	rref	减缩行格式矩阵
	subspace	两空间夹角
● 线性方程		
	\ 和 /	线性方程求解
	inv	矩阵求逆

续表

● 线性方程		
	Rcond	Linpack 逆条件数估计
	cond	计算矩阵的条件数
	condest	范数为 1 的条件数估计
	normest1	1 范数估计
	chol	Cholesky 因式分解
	cholinc	不完全的 Cholesky 因式分解
	lu	LU 因式分解
	luinc	不完全的 LU 因式分解
	qr	正交三角矩阵分解
	lsqnonneg	非负的线性最小二乘法求解
	pinv	矩阵伪逆
	lscov	协方差已知的最小二乘求解
● 特征值与奇异值		
	eig	求特征值和特征向量
	svd	奇异值分解
	gsvd	广义奇异值分解
	eigs	几个特征值
	svds	几个奇异值
	poly	求特征多项式
	polyeig	多项式特征值问题
	condeig	关于特征值的条件数
	hess	Hessberg 形式
	qz	广义特征值
	schur	Schur 分解
● 矩阵函数		
	expm	矩阵指数
	expm1	实现 expm 的 M 文件
	expm2	通过泰勒级数求矩阵指数
	expm3	通过特征值和特征向量求矩阵指数
	logm	矩阵对数
	sqrtm	矩阵开平方根
	funm	一般矩阵的计算

表 A.15　运算符与特殊字符

- 算术运算符

运算符	说明
+	加或正号
-	减或符号
*	矩阵乘法
.*	向量乘法
^	矩阵幂
.^	向量幂
\	左除或反斜杠
/	右除或斜杠
./	向量右除
.\	向量左除
kron	Kronecker 张量积

- 关系运算符

运算符	说明
==	等于
~=	不等于
<	小于
>	大于
<=	小于等于
>=	大于等于

- 逻辑运算符

<table>
<tr><td>&</td><td>逻辑与</td></tr>
<tr><td>|</td><td>逻辑或</td></tr>
<tr><td>~</td><td>逻辑非</td></tr>
<tr><td>XOR</td><td>逻辑异或</td></tr>
<tr><td>any</td><td>向量任一元为真，则其值为真</td></tr>
<tr><td>all</td><td>向量所有元为真，则其值为真</td></tr>
</table>

- 特殊字符

字符	说明
:	冒号
( )	圆括号
[ ]	方括号
{ }	曲柄与下标
@	建立函数句柄
.	小数点
..	父目录
...	继续
,	逗号

续表

● 特殊字符		
	;	分号
	%	注释
	!	感叹号
	=	赋值
	'	复共轭
	.'	转置
	[,]	水平串联
	[;]	垂直串联
	(),{},.	下标赋值或索引
● 位运算符		
	bitand	位与
	bitcmp	位余
	bitor	位或
	bitmax	最大浮点整数
	bitxor	位异或
	bitset	设置位
	bitget	获取位
	bitshift	位移位
● 设置运算符		
	union	设置并集
	unique	设置独立集
	intersect	设置交集
	setdiff	设置微分
	setxor	设置异或
	ismember	为设置对象返回 TRUE

表 A.16　多项式与内插函数

● 数据内插		
	pchip	分段立方 Hermite 内插多项式
	interp1	一维数据内插
	interp1q	快速一维线性内插
	interpft	利用 fft 进行一维数据内插
	interp2	二维数据内插
	interp3	三维数据内插
	interpn	N 维数据内插
	griddata	数据网格

续表

● 数据内插		
	griddata3	三维数据网格
	griddatan	N 维数据网格
● 样条内插		
	spline	3 次样条数据内插
	ppval	分段多项式计算
● 多项式		
	roots	求多项式的根
	poly	构造具有指定根的多项式
	polyval	多项式计算
	polyvalm	带矩阵变量的多项式计算
	residue	部分分式展开
	polyfit	数据的多项式拟合
	polyder	微分多项式
	polyint	积分多项式分析
	conv	多项式乘法
	deconv	多项式除法

表 A.17 稀疏矩阵函数

● 基本稀疏矩阵		
	speye	单位稀疏矩阵
	sprand	均匀分布的随机稀疏矩阵
	sprandn	正态分布的随机稀疏矩阵
	sprandsym	对称的随机稀疏矩阵
	spdiags	从对角阵中形成稀疏矩阵
● 完全矩阵与稀疏矩阵间变换		
	sparse	由非零元素及其序号形成稀疏矩阵
	Full	变稀疏矩阵为完全矩阵
	find	找出非零元素的序号
	spconvert	稀疏矩阵外部结构的变换
● 稀疏矩阵非零元素的处理		
	Nnz	非零元素的数目
	nonzeros	非零元素
	nzmax	分配给非零元素的存储量
	spones	用“1”取代非零元素
	spalloc	为非零元素分配内存
	issparse	若为稀疏矩阵返回 TRUE

续表

● 稀疏矩阵非零元素的处理		
	Spfun	只对非零元素取函数
	Spy	显示稀疏矩阵
● 排序算法		
	colamd	列近似最小度
	symamd	最小近似对称度
	colmmd	列最小度
	symmmd	最小对称度
	symrcm	逆 Cathill—Mckee 序
	colperm	基于非零元素按列排列
	randperm	随机排列向量
	dmperm	Dulmage-Mendelsohn 分解
● 线性几何与迭代方法		
	eigs	多个特征值
	svds	多个奇异值
	luinc	不完全的 LU 分解
	cholinc	不完全的 Cholesky 分解
	normest	矩阵 2 范数估计
	condest	范数 1 的条件值估计
	sprank	结构秩
	Pcg	前提变化梯度法
	bicg	二次梯度变化法
	bicgstab	二次稳定梯度变化法
	Cgs	变化梯度平方法
	gmres	最小残余归纳法
	minres	最小留数法
	Qmr	准最小留数法
	symmlq	对称 LQ 法
● 树型操作		
	colamdtree	后序列消元树的列近似最小度
	symamdtree	后序列消元树的最小近似对称度
	treelayout	显示一个或多个结构树
	treeplot	画结构树
	etree	求矩阵的消元树
	etreeplot	画消元树图
	gplot	绘图

续表

● 其他

	symbfact	符号分解分析
	spparms	为稀疏矩阵处理过程设置参数
	spaugment	形成最小二乘增广系统

表 A.18 特殊数学函数

● 特殊数学函数

	airy	Airy 函数
	besselj	第一类 Bessel 函数
	bessely	第二类 Bessel 函数
	besselh	第三类 Bessel 函数
	besseli	改进的第一类 Bessel 函数
	besselk	改进的第二类 Bessel 函数
	beta	β 类函数
	betainc	不完全的 β 函数
	betaln	β 函数的对数
	ellipj	雅可比椭圆函数
	ellipke	完全椭圆积分
	erf	误差函数
	erfc	互补误差函数
	erfcx	比例互补误差函数
	erfinv	逆误差函数
	expint	指数积分函数
	gamma	γ 函数
	gammainc	不完全 γ 函数
	gammaln	γ 函数的对数
	legendre	联合的 Legendre 函数
	cross	向量的叉乘
	dot	向量的点乘

● 数字理论函数

	factor	主要参数
	isprime	若为主要参数返回 TRUE
	gcd	求最大公约数
	lcm	求最小公倍数
	rat	有理逼近
	rats	有理输出
	perms	所有可能的转置

续表

● 坐标变换		
	cart2sph	变卡笛尔坐标为球坐标
	cart2pol	变卡笛尔坐标为极坐标
	pol2cart	变极坐标为卡笛尔坐标
	sph2cart	变球坐标为卡笛尔坐标
● 其他		
	Besschk	检查的 Bessel 函数参数
	betacore	不完全 β 函数的中心算法
	erfcore	误差函数的中心算法

表 A.19 特殊图形函数

	Area	填充绘图区域
	Bar	条形图
	barh	水平条形图
	Bar3	三维条形图
	Bar3h	三维水平条形图
	comet	星点图
	comet3	三维星点图
	compass	区域图
	contour	等高线图
	contourf	填充等高线图
	contour3	三维等高线图
	errorbar	误差条形图
	feather	箭头图
	hist	直方图
	Pie	二维饼图
	Pie3	三维饼图
	plotmatrix	矩阵的散点图
	rose	角度直方图
	scatter	散点图
	stem	离散序列图或杆图
	stairs	阶梯图
	surf	三维曲面阴影图
	surfc	曲面和等高线混合图
	surfl	带亮度的三维曲面阴影图
	waterfall	落差图

表 A.20 字符串函数

● 一般函数		
	char	产生字符串
	double	将字符串转换成数字特征码
	cellstr	由字符数组产生单元数组
	blanks	空串
	deblanks	删除尾部的空串
	eval	执行由 MATLAB 表达式组成的串
● 字符串测试		
	ischar	若为字符串返回 TRUE
	iscellstr	若为单元串返回 TRUE
	isletter	若为字母表中的字母返回 TRUE
	isspace	若为空格返回 TRUE
● 字符串比较		
	strcat	连接串
	strvcat	垂直连接串
	strcmp	比较字符串
	strncmp	串中前 N 个字符比较
	findstr	在字符串中查找子串
	strjust	串对齐
	strmatch	寻找匹配串
	strrep	取代字符串
	strtok	在字符串中查找标记
	upper	变字符串为大写
	lower	变字符串为小写
● 字符串与数值的转换		
	Num2str	变数值为字符串
	Int2str	变整数为字符串
	mat2str	把数值数组转换成输入形态串数组
	str2double	字符串转换为双精度数值
	str2num	变字符串为数值
	sprintf	变数值为格式控制下的字符串
	sscanf	变字符串为格式控制下的数值
● 数值之间的变换		
	hex2num	变十六进制数为 IEEE 标准的浮点数
	hex2dec	变十六进制数为十进制数
	dec2hex	变十进制数为十六进制数

续表

● 数值之间的变换		
	bin2dec	变二进制数为十进制数
	dec2bin	变十进制数为二进制数
	base2dec	变 X 进制数为十进制数
	dec2base	变十进制数为 X 进制数

表 A. 21　时间与日期函数

Now	现在的时间与日期作为日期数
date	现在的日期作为日期数
clock	现在的时间与日期作为日期向量
datenum	连续的日期数
datestr	日期的串表示
datevec	日期成分
calendar	日历
weekday	周日
eomday	月底
datetick	格式化的日期
cputime	CPU 时间
Tic	开始秒表记时
Toc	秒表停止
etime	记时函数
Pause	以秒为单位等待

表 A. 22　图形用户界面工具

uicontrol	创建用户界面控制
Uimenu	创建用户界面菜单
Ginput	用鼠标输入图形
dragrect	鼠标拖动画 XOR 矩形
rbbox	涂抹块
selectmoveresize	交互式选择、移动、调整或复制对象
waitforbuttonpress	在图形中等待按键/按钮
waitfor	块执行与等待事件
uiwait	块执行与等待继续
uiresume	继续执行块 M 文件
uistack	控制对象的堆栈次序
uisuspend	延缓图形的交互式状态
uirestore	恢复图形的交互式状态

续表

Guide	设计图形用户界面
inspect	检查目标属性
align	将用户控制界面与坐标轴对齐
propedit	编辑属性
axlimdlg	轴限制对话框
dialog	建立对话框
errordlg	建立出错对话框
helpdlg	建立帮助对话框
imageview	利用放大图形显示图像
inputdlg	建立输入对话框
listdlg	列出选择对话框
menu	建立用户输入选择菜单
movieview	用重放在图形中观看动画
msgbox	建立消息框
pagedlg	建立页对话框
pagesetupdlg	页设置对话框
printdlg	打印对话框
printpreview	打印预览
questdlg	建立提问对话框
uigetpref	具有优先权的提问对话框
soundview	在图形中播放声音文件
uigetfile	标准的打开文件对话框
uiputfile	标准的保存文件对话框
uisetcolor	颜色选择对话框
uisetfont	字体设置对话框
uiopen	显示打开文件对话框并打开文件
uisave	显示保存文件对话框并保存文件
uiload	显示打开文件对话框并装载文件
uiimport	对输入数据开始图形用户界面
waitbar	显示等待条
warndlg	警告对话框

表 A.23　版本控制

checkin	将文件记入版本控制系统
checkout	校验来自版本控制系统的文件
undocheckout	取消校验来自版本控制系统的文件
Rcs	用 RCS 控制版本行为

续表

	Pvcs	用 PVCS 控制版本行为
	clearcase	用 ClearCase 控制版本行为
	sourcesafe	用 Visual SourceSafe 控制版本行为
	customverctrl	用户版本控制模板

表 A.24 Windows 操作系统文件界面

	winfun	控件类
	ddeadv	设置咨询链接
	ddeexec	发送串执行命令
	ddeinit	开始 DDE
	ddepoke	给应用发送数据
	ddereq	向应用要求数据
	ddeterm	终止 DDE
	ddeunadv	释放咨询链接

附录 B　Toolbox函数

MATLAB 6 中共有 40 多个工具箱，包括了通讯、控制系统、信号处理、图像处理、神经网络及模糊系统等众多领域，而且工具箱还在不断增加，这些工具箱给各个领域的研究和工程应用提供了强有力的工具。为了便于用户查阅，这里列出了 10 种基本的工具箱函数。

表 B.1 为本附录所列出的 Toolbox 工具箱。表 B.2～表 B.10 为各个 Toolbox 工具箱中所提供的函数及其用途。

表 B. 1　MATLAB 6 的部分 Toolbox

目录名称	工具箱名称	索　引
control	控制系统	表 B.2
fuzzy	模糊逻辑	表 B.3
ident	系统辨识	表 B.4
images	图像处理	表 B.5
nnet	神经网络	表 B.6
optim	最优化	表 B.7
robust	鲁棒控制	表 B.8
signal	信号处理	表 B.9
wavelet	小波	表 B.10

表 B. 2　控制系统工具箱

● 一般		
	ctrlpref	设置控制系统工具箱的参数
	ltimodels	关于 LTI 系统各种形式的详细帮助
	ltiprops	关于 LTI 系统特性的详细帮助
● 建立线性模型		
	Tf	建立转移函数模型
	Zpk	建立零极点增益模型
	Ss	建立状态空间模型
	Dss	建立描述状态空间模型
	Frd	建立频率响应模型
	Filt	指定数字滤波器
	Set	设置或调整 LTI 模型的特性

续表

● 数据提取		
	tfdata	提取分子与分母
	zpkdata	提取零极点与增益
	ssdata	提取状态空间矩阵
	dssdata	状态空间矩阵的描述版本
	frdata	提取频率响应数据
	get	获取 LTI 模型值
● 变换		
	Tf	变为转移函数模型
	zpk	变为零极点增益模型
	Ss	变为状态空间模型
	frd	变为频率数据
	chguints	改变频响的频率单位
	c2d	连续系统到离散系统的转换
	d2c	离散系统到连续系统的转换
	d2d	重采样离散时间模型
● 系统互连		
	append	通过附加输入与输出组合 LTI 系统
	parallel	广义并行连接
	series	广义串行连接
	feedback	两个系统的反馈连接
	lft	广义反馈互连
	connect	从结构图中获得状态空间模型
● 动态模型		
	dcgain	直流增益
	pole	系统极点
	zero	系统零点
	pzmap	零极图
	damp	阻尼系数与固有频率
	esort	按实部排序连续特征值
	dsort	按幅值排序离散特征值
	norm	LTI 系统的范数
	covar	相对于白噪声的协方差响应
● 时域分析		
	ltiveiw	LTI 浏览器
	step	阶跃响应

续表

● 时域分析		
	Impulse	单位抽样响应
	initial	给定初始状态的状态空间系统响应
	lsim	任意输入下的响应
	gensig	为 LSIM 产生输入
● 频域分析		
	bode	频率响应的波特图
	bodemag	波特幅度图
	sigma	奇异值频域图
	nyquist	Nyquist 图
	nichols	Nichols 图
	margin	增益与相位裕度
	allmargin	全频率与增益或相位裕度
	freqresp	频率栅格表示的频率响应
	evalfr	给定频率点的频率响应
	interp	插入频率响应数据
● 极点配置		
	Place	多输入多输出极点配置
	acker	单输入单输出极点配置
	extim	形成给定增益的估计器
	Reg	形成给定状态反馈与增益的调节器
● 系统设计		
	sisotool	单输入单输出用户界面
	rlocus	Evens 根轨迹
	lqr	线性二次状态反馈调节器设计
	lqry	输出加权的 LQ 调节器设计
	lqrd	连续代价函数离散 LQ 调节器设计
	kalman	Kalman 估计器
	kalmd	连续代价函数的离散 Kalman 估计器
	lqgreg	形成 LQG 调节器与 Kalman 估计器
	augstate	通过附加状态增加输出
● 状态空间模型		
	rss	随机稳定状态空间模型
	ss2ss	状态相似变换
	canon	状态空间的正则形式
	Ctrb	可控性矩阵

续表

● 状态空间模型		
	Obsv	可观性矩阵
	gram	可控性与可观性
	ssbal	实现状态空间的对角平衡
	balreal	基于 Gramian 的输入输出平衡
	modred	模型降阶
	mineral	最小实现与零极点对消
	sminreal	结构最小实现
● 时延		
	hasdelay	若模型具有时延返回 TRUE
	totaldelay	一个输入/输出对的总延时
	pade	时延的 Pade 近似
● 模型维数与特性		
	class	模型类型
	Isa	测试模型是否属于给定类型
	size	模型的大小与阶数
	ndims	维数
	isempty	若模型为空返回 TRUE
	isct	若为连续时间模型返回 TRUE
	isdc	若为离散时间模型返回 TRUE
	isproper	若模型正确返回 TRUE
	issiso	若为单输入单输出返回 TRUE
	reshape	调整线性模型的向量
● 重载算术操作		
	+ 与 -	系统的加与减
	*	系统相乘
	\ 与 /	系统的左除与右除
	^	给定系统的幂
	.'	输入输出图的转置
	[• •]	模型连接
	stack	沿数组的某维堆叠模型/数组
	Inv	LTI 系统的逆
● 求解矩阵方程		
	lyap	连续 Lyapunov 方程求解
	dlyap	离散 Lyapunov 方程求解
	care	连续 Riccati 方程求解

续表

● 求解矩阵方程		
	dare	离散 Riccati 方程求解

表 B.3 模糊逻辑工具箱

● GUI 编辑器		
	Anfisedit	ANFIS 训练与测试用户界面工具
	findcluster	聚类用户界面工具
	Fuzzy	基本 FIS 编辑器
	mfedit	隶属度函数编辑器
	ruleedit	规则编辑器及句法分析程序
	releview	规则观察器及模糊推理框图
	surfview	输出曲面观测器
● 隶属度函数		
	dsigmf	两个"S"形隶属度函数的差
	gauss2mf	双边高斯曲线隶属度函数
	gaussmf	高斯曲线隶属度函数
	gbellmf	广义钟形隶属度函数
	pimf	π 形隶属度函数
	psigmf	两个"S"形隶属度函数的积
	smf	"S"形隶属度函数
	sigmf	"sigmoid" 形隶属度函数
	trapmf	梯形隶属度函数
	trimf	三角形隶属度函数
	zmf	"Z"形隶属度函数
● 命令行 FIS 函数		
	addmf	将隶属度函数加到 FIS 中
	addrule	将规则加到 FIS 中
	addvar	将变量加到 FIS 中
	defuzz	去模糊隶属度函数
	evalfis	完成模糊推理运算
	evalmf	隶属度函数计算
	gensurf	产生 FIS 输出曲面
	getfis	获得模糊系统的特性
	mf2mf	在函数间变换参数
	newfis	产生新的 FIS
	parsrule	分析模糊规则
	plotfis	显示 FIS 输入/输出图

续表

● 命令行 FIS 函数		
	Plotmf	显示一个变量的所有隶属度函数
	readfis	从磁盘中装入 FIS
	rmmf	从 FIS 中删除隶属度函数
	rmvar	从 FIS 中删除变量
	setfis	设置模糊系统特性
	showfis	显示带注释的 FIS
	showrule	显示 FIS 规则
	writefis	在磁盘中保存 FIS
● 高级技术		
	anfis	Sugeno—type FIS 的训练程序
	fcm	利用模糊 C 平均聚类法找出簇
	genfis1	利用一般方法产生 FIS 矩阵
	genfis2	利用减法聚类法产生 FIS 矩阵
	subclust	利用减法聚类法估计簇中心
● 其他函数		
	discfis	离散一个模糊推理系统
	evalmmf	为多个成员函数赋值
	fstrvcat	连接变长度的矩阵
	fuzaritn	模糊代数函数
	findrow	找出与输入串匹配的矩阵行
	genparam	产生 ANFIS 学习的初始参数
	probor	可行性 OR
	sugmax	一个 Sugeno 系统的最大输出范围

表 B.4 系统辨识工具箱

● 仿真与预测		
	Predict	M 步超前预测
	Pe	计算预测误差
	Sim	仿真一给定系统
● 数据处理		
	Iddata	构造一数据对象
	detrend	从数据集中删除方位
	Idfilt	用 Butterworth 滤波器对数据滤波
	idinput	对识别系统产生输入信号
	Merge	合并几个实验
	misdata	估计并代替丢失的输入输出数据

续表

● 数据处理		
	resample	利用抽样或内插重新采样数据
● 非参数估计		
	Covf	估计数据矩阵的协方差矩阵
	Cra	相关分析
	etfe	估计经验传递函数并计算周期图
	impulse	估计单位抽样响应
	spa	频谱分析
	step	阶跃响应
● 参数估计		
	Ar	利用各种方法的 AR 信号模型
	armax	ARMAX 模型预测误差估计
	arx	ARX 模型的最小二乘估计
	bj	Box—Jenkins 模型的预测误差估计
	ivar	时间序列的 AR 部分的仪器 IV 估计
	iv4	ARX 模型近似最优的 IV 估计
	N4sid	采用子空间方法估计状态空间模型
	oe	输出误差模型的预测误差估计
	pem	一般线性模型的预测误差估计
● 建立模型结构		
	idploy	从给定多项式构造模型对象
	idss	构造状态空间模型对象
	idarx	构造多变量 ARX 模型对象
	idgrey	构造用户参数化模型对象
● 模型变换		
	arxdata	将模型转换成 ARX 矩阵
	polydata	与给定模型相关的多项式
	ssdata	提取状态空间矩阵
	tfdata	提取分子与分母
	zpkdata	提取零极点与增益
	idfrd	模型的频率函数
	idmodred	降低模型的阶数
	C2d	连续系统到离散系统的转换
	D2c	离散系统到连续系统的转换
● 模型表示		
	bode	频率响应的波特图

续表

● 模型表示		
	ffplot	频率函数图
	plot	数据对象的输入输出图
	present	屏幕上的参数模型
	pzmap	零极图
	nyquist	Nyquist 图
	view	LTI 浏览器
● 模型合法化		
	compare	将仿真和预测输出与测量输出比较
	pe	预测误差
	predict	M 步超前预测
	resid	计算和测试与某模型相关的留数
	sim	仿真一给定的系统
● 模型结构选择		
	Arxstruc	ARX 模型类的损失函数
	selstruc	根据各种准则选择模型结构
	Struc	arxstruc 的典型结构矩阵
● 递归参数估计		
	Rarx	对 AR 模型递归计算估值
	Rarmax	对 ARMAX 模型递归计算估值
	Rbj	对 Box—Jenkins 模型递归计算估值
	Roe	对输出误差模型递归计算估值
	Rpem	对一般模型递归计算估值
	Rplr	对一般模型递归计算估值
	segment	分段数据并跟踪快变系统

表 B.5 图像处理工具箱

● 图像显示		
	colorbar	显示颜色条
	truesize	改变图像大小使之具有实际尺寸
	warpgetimage	将图像卷成曲面获取图像
	image	创建并显示图像
	imagesc	数据定标并按图像显示
	immovie	制作图像动画
	imshow	显示所有类型的图像数据
	montage	按矩形剪辑方式显示图像
	subimage	显示多个图像

续表

● 图像文件 I/O		
	Imfinfo	返回有关图像文件的信息
	Imread	读图像文件
	imwrite	写图像文件
● 几何操作		
	Imcrop	修剪图像
	imresize	改变图像大小
	imrotate	旋转图像
	Interp2	二维数据内插
● 像素值与统计		
	corr2	二维相关系数
	imcontour	图像等高线
	imfeature	计算图像的特征度
	Imhist	图像的直方图
	impixel	决定像素的颜色值
	improfile	图像轮廓强度
	mean2	矩阵的均值
	Pixval	有关图像的像素信息
	std2	二维标准差
● 图像分析		
	Edge	边界提取
	qtdecomp	进行 QT 分解
	qtgetblk	获取 QT 分解的块值
	qtsetblk	设置 QT 分解的块值
● 图像增强		
	Histeq	直方图均衡化
	imadjust	调整与展宽图像强度
	Imnoise	给图像增加噪声
	Medfilt2	二维中值滤波
	ordfilt2	二维阶统计滤波
	Wiener2	自适应二维维纳滤波
● 线性滤波		
	conv2	二维卷积
	convmtx2	二维矩阵卷积
	Convn	N 维卷积
	Filter2	二维滤波

续表

● 线性滤波		
	fspecial	特殊的二维滤波
	freqspace	二维频率响应
	Freqz2	二维数字频率响应
	Fsamp2	频率采样的二维 FIR 滤波器设计
	ftrans2	频率变换的二维 FIR 滤波器设计
	Fwind1	一维窗函数的 FIR 滤波器设计
	Fwind2	二维窗函数的 FIR 滤波器设计
● 图像变换		
	dct2	二维离散余弦变换
	Dctmtx	计算离散余弦变换矩阵
	fft2	二维快速傅立叶变换
	Fftn	N 维快速傅立叶变换
	fftshift	零频移到频谱中心
	idct2	二维逆离散余弦变换
	ifft2	二维逆快速傅立叶变换
	Ifftn	N 维逆快速傅立叶变换
	Iradon	逆 Radon 变换
	phantom	产生头幻影图像
	Radon	Radon 变换
● 块操作		
	bestblk	分块处理的块大小
	blkproc	按块处理图像
	col2im	重新排列以形成图像
	colfilt	局部非线性滤波
	im2col	重新排列成列
	nlfilter	局部非线性滤波
● 二进制图像处理		
	applylut	用查寻表进行近邻操作
	Bwarea	二进制图像中的目标区域
	bweuler	欧拉数
	Bwfill	填充二进制图像的背景
	bwlabel	标注二进制图像的相关部分
	bwmorph	形态算子
	bwperim	二进制图像中目标的周围
	bwselect	选择二进制图像中的对象

续表

● 二进制图像处理		
	Dilate	加浓二进制图像
	Erode	冲淡二进制图像
	makelut	用 applylut 构造查寻表
● 个别区域		
	roicolor	用颜色定义感兴趣的区域
	roifill	在任意区域平稳地改写
	roifilt2	对感兴趣的区域滤波
	roipoly	定义感兴趣的多边区域
● 颜色操作		
	brighten	加亮或增暗颜色板
	cmunique	寻找唯一的颜色板及相应的图像
	cmpermute	置换颜色板位置
	colormap	设置或获取颜色查寻表
	hsv2rgb	变 HSV 值为 RGB 颜色空间
	imapprox	利用更少颜色的图像逼近加标图像
	ntsc2rgb	变 NTSC 值为 RGB 颜色空间
	rgb2hsv	变 RGB 值为 HSV 颜色空间
	rgb2ntsc	变 RGB 值为 NTSC 颜色空间
	rgbplot	绘制 RGB 颜色板分量的图形
	rgb2ycbcr	变 RGB 值为 YCBCR 颜色空间
	ycbcr2rgb	变 YCBCR 值为 RGB 颜色空间
● 图像类型与类型变换		
	Dither	Floyd—Steinberg 图像颤抖
	gray2ind	变灰度图像为附标图像
	grayslice	阈值处理从灰度图像产生附标图像
	im2bw	变图像为黑白图形
	im2double	变图像矩阵为双精度形式
	im2unit8	变图像矩阵为无符号 8 位整数形式
	im2unit16	变图像矩阵为无符号 16 位整数形式
	ind2gray	变附标图像为灰度图像
	ind2rgb	变附标图像为 RGB 图像
	mat2gray	变矩阵为灰度图像
	Isbw	若为二进制图像返回 TRUE
	Isgray	若为灰度图像返回 TRUE
	Isind	若为附标图像返回 TRUE

续表

● 图像类型与类型变换		
	Isrgb	若为 RGB 图像返回 TRUE
	rgb2gray	变 RGB 图像为灰度图像
	rgb2ind	变 RGB 图像为附标图像
	rgb2hsv	变 RGB 图像为 HSV 图像
	rgb2ntsc	变 RGB 图像为 NTSC 图像

表 B.6 神经网络工具箱

● 分析函数		
	Errsurf	计算误差曲面
	maxlinlr	线性层的最大学习率
● 距离函数		
	Boxdist	盒距离函数
	Dist	欧几里得距离函数
	mandist	曼哈顿距离函数
	linkdist	链距离函数
● 初始化函数		
	Initnw	Nguyen—Widow 层初始化函数
	Initlay	层间初始化函数
	Initcon	偏差初始化函数
	Initzero	零加权/偏差初始化函数
	Midpoint	中点加权初始化函数
	Randnc	归一化列加权初始化函数
	Randnr	归一化行加权初始化函数
	Rands	对称随机加权/偏差初始化函数
● 学习函数		
	learncon	偏差学习函数
	learngd	梯度下降加权/偏差学习函数
	learngdm	梯度下降要素加权/偏差学习函数
	learnh	Hebb 学习规则
	learnhd	退化的 Hebb 学习规则
	learnis	内星学习规则
	learnk	Kohonen 学习规则
	learnlv1	LVQ1 加权学习函数
	learnlv2	LVQ2 加权学习函数
	learnos	外星学习规则
	learnp	感知层学习规则

续表

● 学习函数		
	Learnpn	归一化的感知层学习规则
	learnsom	自组织学习规则
	learnwh	Widrow—Hoff 学习规则
● 搜索函数		
	srchbac	反向搜索
	srchbre	Brent 联合黄金分割/二次插值
	srchcha	Charalambous 三次插值
	srchgol	黄金分割搜索
	srchhyb	Hybird 对分/三次搜索
● 新网络		
	network	建立用户神经网络
	Newc	建立竞争层
	newcf	建立前向层叠 BP 网
	newelm	建立 Elman 前向 BP 网
	newff	建立前馈 BP 网
	newfftd	建立前馈输入延时 BP 网
	newgrnn	建立广义递归网
	newhop	建立 Hopfield 循环网
	newlin	建立线性层
	newlind	设计线性层
	newlvq	建立学习向量量化网
	Newp	建立感知器
	newpnn	建立概率神经网络
	newrb	设计半径基网络
	newrbe	设计精确的半径基网络
	newsom	建立自组织图
● 网络输入函数		
	netprod	网络输入函数的乘积
	netsum	网络输入函数的和
	dnetprod	网络输入衍生函数的乘积
	dnetsum	网络输入衍生函数的和
● 执行函数		
	Mae	平均绝对误差执行函数
	Mse	均方差执行函数
	msereg	具有规则的均方差执行函数

续表

● 执行函数		
	Sse	和方差执行函数
	Dmae	平均绝对误差执行衍生函数
	Dmse	均方差执行衍生函数
	dmsereg	具有规则的均方差执行衍生函数
	Dsse	和方差执行衍生函数
● 绘图函数		
	hintonw	绘制权值图
	hintonwb	绘制权值与偏差图
	plotbr	按贝叶斯规则训练的网络执行图
	plotes	绘制单神经元的误差曲面
	Plotpc	在感知器矢量图中画出分类线
	plotpv	画出感知器输入/目标向量
	plotep	在误差曲面上画出加权偏差位置
	plotperf	绘制网络执行图
	plotsom	绘制自组织图
	plotv	从原点绘制矢量线
	plotvec	用不同颜色绘制矢量
● 预处理与后处理		
	prestd	单元标准偏差与零均值归一化数据
	poststd	用 prestd 来归一化数据
	trastd	转换已计算均值与标准差的数据
	premnmx	将数据归一化到[-1 1]区间内
	postmnmx	用 premnmx 来归一化数据
	tramnmx	转换已计算最小值与最大值的数据
	prepca	对输入数据进行成分分析
	trapca	转换用 prepca 计算的 PCA 数据
	postreg	后向训练递归分析
● 拓扑函数		
	gridtop	绘制拓扑函数的栅格层
	hextop	六边形拓扑函数
	randtop	随机拓扑函数
● 训练函数		
	trainb	用加权与偏差学习规则进行批训练
	trainbfg	BFGS 准 Newton 后向传播
	Trainbr	贝叶斯规则

续表

● 训练函数		
	Trainc	训练竞争层网络
	traincgb	Powell—Beale 共轭梯度后向传播
	traincgf	Fletcher—Powell 共轭梯度后向传播
	traincgp	Polak—Ribiere 共轭梯度后向传播
	Traingd	梯度下降后向传播
	traingdm	具有动力的梯度下降后向传播
	Traingda	具有自适应 LR 的梯度下降后向传播
	Traingdx	自适应 LR 与动力梯度下降后向传播
	Trainlm	Levenberg—Marquardt 后向传播
	trainoss	一步交叉后向传播
	trainr	随机阶数增加训练与学习函数
	trainrp	弹回后向传播
	Trains	连续阶数增加训练/学习函数
	Trainseg	共轭梯度后向传播
● 传递函数		
	Compet	竞争层传递函数
	Hardlim	硬限幅传递函数
	Hardlims	对称硬限幅传递函数
	logsig	对称 S 型传递函数
	poslin	正线性传递函数
	purelin	线性传递函数
	radbas	径向基传递函数
	satlin	饱和线性传递函数
	satlins	对称饱和线性传递函数
	softmax	软最大传递函数
	tansig	正切 S 型传递函数
	tribas	三角基传递函数
● 应用网络		
	Sim	仿真神经网络
	Init	初始化神经网络
	Adapt	自适应神经网络
	Train	训练神经网络
	Disp	显示神经网络的特性
	display	显示神经网络变量的名字与特性

续表

● 向量与矩阵		
	cell2mat	将单元向量联合成矩阵
	Concur	创建并行偏差向量
	con2seg	将并行向量转换成串行向量
	combvec	创建所有的矢量集
	ind2vec	变下标向量为稀疏矩阵表示
	mat2cell	将矩阵分成单元向量
	Minmax	矩阵的行距
	Nncopy	拷贝矩阵或单元向量
	Normc	归一化矩阵列
	Normr	归一化矩阵行
	Pnormc	伪归一化矩阵列
	Quant	离散化成某数值的整数倍
	seq2com	将串行向量转换成并行向量
	Sumsqr	平方和
	vec2ind	变稀疏矩阵表示为下标向量
● 权函数		
	Dist	欧几里得距离函数
	Dotprod	点积权函数
	Mandist	曼哈顿距离函数
	Negdist	点积权函数
	Normprod	归一化的点积权函数

表 B.7 最优化工具箱

● 矩阵问题的最小化		
	linprog	线性规划
	quadprog	二次规划
● 线性最小二乘		
	lsqlin	具有约束条件的线性二乘
	lsqnonneg	具有非负条件的线性二乘
● 非线性最小化函数		
	fminbnd	具有约束条件的非线性最小化
	fmincon	多维约束非线性最小化
	fminsearch	多维无约束非线性最小化
	fminunc	多维无约束非线性最小化
	fseminf	多维半定约束非线性最小化
	fgoalattain	多维即定目标最优化

续表

● 非线性最小化函数		
	Fminimax	多维极小极大最优化
	lsqcurvefit	非线性曲线拟合
	lsqnonlin	具有上下边界的非线性最小二乘
	fzero	非线性寻找零点
	fsolve	解非线性系统方程
● 控制默认值与选项		
	optimset	创建或改变最优化选项
	optimget	获取最优化选项
● 演示		
	circustent	二次规划
	molecule	用约束非线性最小化构造解
	optdeblur	采用约束线性二乘的图像去污点
	optdemo	演示菜单
	tutdemo	启动教程
	banddemo	香蕉型函数的最小化
	goaldemo	目标达到
	dfildemo	有限精度滤波器设计
	datdemo	数据拟合曲线
● 三次内插程序		
	cubic	内插 4 点以找出最大值
	cubici1	内插 2 点和梯度，以估计最小值
	cubici2	内插 3 点和 1 梯度
	cubici3	内插 2 点和梯度以找出步长与最小值
● 二次内插程序		
	quad2	内插 3 点以找出最大值
	quadi	内插 3 点以估计最小值
● 演示实用程序		
	eigfun	返回分类特征值的函数
	elimone	消去一变量
	filtobj	频响范数
	fitfun	返回逆合数据中的误差范数
	filtcon	频响根
	fitfun2	返回逆合数据中的误差矢量
	tentdata	circustent 演示数据的 MAT 文件
	optdeblur	optdeblur 演示数据的 MAT 文件

续表

● 演示实用程序

molecule	molecule 演示数据的 MAT 文件
Mmole	molecule 演示的距离问题
bandem	香蕉型函数最小化演示
optdems	设置最优化演示
banplot	画出一步解路径
banplot2	画出一步解路径

● 半定实用程序

semifun	半定问题转换成约束问题
semicon	半定约束转换成约束问题
findmax	在数据向量中内插极大值
findmax2	在数据矩阵中内插极大值
v2sort	分类两向量并删去丢失的元素

● 目标达到的实用程序

goalfun	目标达到问题转换成约束条件问题
goldcon	改变目标达到问题中的约束条件

● 其他

Color	对稀疏有限差分进行列分割
graderr	用于检查梯度的不一致性
Lsint	初始化最小二乘程序的函数
optint	初始化无约束最小化程序的函数
searchq	线性搜索程序
Sfd	有限梯度差分
sfdnls	有限差分
updhess	进行逆 Hessian 修正

表 B.8 鲁棒控制工具箱

● 可选系数数据结构

branch	从树中提取一分支
graft	从树中增加一分支
issystem	识别一系统变量
istree	识别一树型变量
mksys	为系统建立树变量
tree	建立树变量
vrsys	返回标准系统变量名

● 建模

augss	状态空间模型的系统增广

续表

● 建模		
	augtf	传递函数模型的系统增广
	interc	一般多变量内连系统
● 模型转换		
	bilin	多变量双线性变换
	Des2ss	奇异值分解变系统为状态空间系统
	Lftf	线性分式变换
	sectf	扇形变换
	stabproj	稳定与逆稳定映射
	slowfast	慢/快分解
	Tfm2ss	变传递函数模型为状态空间模型
● 实用工具		
	aresolv	广义连续时间 Riccati 方程求解
	daresolv	广义离散时间 Riccati 方程求解
	riccond	连续时间 Riccati 方程的条件数
	driccond	离散时间 Riccati 方程的条件数
	blkrsch	通过 cschur 得到块有序实 Schur 形式
	cschur	通过复旋转得有序复 Schur 形式
● 多变量 Bode 图		
	cgloci	连续特性增益轨迹
	dcgloci	离散特性增益轨迹
	dsigma	离散奇异值 Bode 图
	muopt	实/复数混合不确定系统的 SSV 上界
	osborne	通过 Osborne 法求得的 SSV 上界
	perron	计算 Perron 特征值
	Psv	Perron 特征结构的 SSV 上界
	sigma	连续奇异值 Bode 图
	ssv	结构化奇异值 Bode 图
● 因式分解技术		
	iofc	内外因子因式分解(行类型)
	iofr	内外因子因式分解(列类型)
	sfl	左边频谱分解
	sfr	右边频谱分解
● 模型简化方法		
	balmr	截断均衡模型简化

续表

● 模型简化方法		
	bstschml	相对误差 Schur 模型简化
	imp2ss	从单位抽样响应到状态空间实现
	obalreal	有序均衡实现
	ohklmr	最优 Hankel 最小化逼近
	rschur	Schur 模型简化
● 鲁棒控制综合方法		
	h2lqg	连续时间 H_2 综合
	Dh2lqg	离散时间 H_2 综合
	hinf	连续时间 H_∞ 综合
	dhinf	离散时间 H_∞ 综合
	hinfopt	H_∞ 综合的 γ 迭代
	normh2	计算 H_2 范数
	normhinf	计算 H_∞ 范数
	lqg	LQG 最优控制综合
	ltru	LQG 闭环传递补偿
	ltry	LQG 闭环传递补偿
	youla	Youla 参数化
● 演示示例		
	accdemo	弹簧质量标准问题
	dintdemo	双积分器系统的 H_∞ 设计
	hinfdemo	飞机或大型空间结构的 H_2 或 H_∞ 设计
	ltrdemo	LQR/LTR 设计
	mudemo	μ 综合示例
	mudemo1	μ 综合示例
	mrdemo	鲁棒模型简化示例
	rctdemo	鲁棒控制工具箱演示

表 B.9 信号处理工具箱

● 滤波器分析与实现		
	abs	幅值
	angle	取相角
	conv	求卷积
	conv2	求二维卷积
	deconv	去卷积
	fftfilt	重叠相加法 FFT 滤波器实现
	filter	直接滤波器实现

续表

● 滤波器分析与实现		
	filter2	二维数字滤波器
	filtfilt	零相位数字滤波
	filtic	filter 初始条件选择
	freqs	模拟滤波器频率响应
	freqspace	频率响应中的频率间隔
	freqz	数字滤波器频率响应
	freqzplot	画出频率响应曲线
	grpdelay	平均滤波延迟
	latcfilt	格形滤波器实现
	impz	数字滤波器的单位抽样响应
	medfilt1	1 维中值滤波
	sgolayfilt	Savitzky—Golay 滤波器实现
	sosfilt	二次分式滤波器实现
	zplane	离散系统零极点图
	upfirdn	上采样
	unwrap	去除相位
● FIR 滤波器设计		
	convmtx	矩阵卷积
	cremez	复、非线性相位等波纹滤波器设计
	fir1	基于窗函数的 FIR 滤波器设计
	fir12	基于窗函数的 FIR 滤波器设计
	fircls	约束的最小二乘滤波器设计
	fircls1	约束的最小二乘 FIR 滤波器设计
	firls	最优最小二乘 FIR 滤波器设计
	firrcos	升余弦滤波器设计
	intfilt	内插 FIR 滤波器设计
	kaiserord	基于阶数估计的凯瑟滤波器设计
	remez	Chebyshev 最优 FIR 滤波器设计
	remezord	基于阶数估计的 remez 设计
	sgolay	Savitzky—Golay FIR 滤波器设计
● IIR 滤波器设计		
	butter	比特沃思滤波器设计
	cheby1	切比雪夫 I 型滤波器设计
	cheby2	切比雪夫 II 型滤波器设计

续表

● IIR 滤波器设计		
	Ellip	椭圆滤波器设计
	maxflat	广义比特沃思低通滤波器设计
	yulewalk	递归数字滤波器设计
	buttord	比特沃思滤波器阶估计
	cheb1ord	切比雪夫 I 型滤波器阶估计
	cheb2ord	切比雪夫 II 型滤波器阶估计
	ellipord	椭圆滤波器阶估计
● 模拟滤波器设计		
	besself	贝塞尔滤波器设计
	butter	比特沃思滤波器设计
	cheby1	切比雪夫 I 型滤波器设计
	cheby2	切比雪夫 II 型滤波器设计
	elip	椭圆滤波器设计
● 模拟滤波器变换		
	lp2bp	低通到带通模拟滤波器变换
	lp2bs	低通到带阻模拟滤波器变换
	lp2hp	低通到高通模拟滤波器变换
	lp2lp	低通到低通模拟滤波器变换
● 滤波器离散化		
	bilinear	双线性变换
	impinvar	冲击响应不变法的模拟到数字变换
● 线性系统变换		
	latc2tf	变格形结构为传递函数形式
	polystab	多项式的稳定性
	polyscale	多项式的根
	residuez	Z 变换部分分式展开
	sos2ss	变二次分式形式为状态空间形式
	sos2tf	变二次分式形式为传递函数形式
	sos2zp	变二次分式形式为零极点增益形式
	ss2sos	变状态空间形式为二次分式形式
	ss2tf	变状态空间形式为传递函数形式
	ss2zp	变状态空间形式为零极点增益形式
	tf2ss	变传递函数形式为状态空间形式
	tf2zp	变传递函数形式为零极点增益形式
	tf2sos	变传递函数形式为二次分式形式

续表

● 线性系统变换		
	tf2latc	变传递函数形式为格形结构
	zp2sos	变零极点增益形式为二次分式形式
	zp2ss	变零极点增益形式为状态空间形式
	zp2tf	变零极点增益形式为传递函数形式
● 窗函数		
	bartlett	Bartlett 窗
	blackman	Blackman 窗
	boxcar	矩形窗
	chebwin	Chebyshev 窗
	hamming	Hamming 窗
	hann	Hanning 窗
	kaiser	Kaiser 窗
	triang	三角窗
● 变换		
	Czt	Chirp z 变换
	Dct	离散余弦变换
	dftmtx	离散傅立叶变换矩阵
	fft	一维快速傅立叶变换
	fft2	二维快速傅立叶变换
	fftshift	重新排列 FFT 的输出
	hilbert	Hilbert 变换
	idct	逆离散余弦变换
	ifft	逆一维快速傅立叶变换
	ifft2	逆二维快速傅立叶变换
● 统计信号处理与谱分析		
	cohere	相关函数平方幅值估计
	corrcoef	相关系数
	corrmtx	相关系数矩阵
	Cov	协方差矩阵
	Csd	互谱密度估计
	pburg	Burg 算法功率谱密度估计
	pcov	协方差法功率谱密度估计
	peig	特征值法功率谱密度估计
	periodogram	周期图法功率谱密度估计
	pmcor	修正协方差法功率谱密度估计

续表

● 统计信号处理与谱分析		
	Pmtm	Thomson 多锥度法功率谱密度估计
	pmusic	Music 法功率谱密度估计
	psdplot	绘制功率谱密度估计
	pyulear	Yule-Walker 法功率谱密度估计
	rooteig	特征值法功率估计
	rootmusic	Music 法功率估计
	Tfe	传递函数估计
	xcorr	一维互相关函数估计
	xcorr2	二维互相关函数估计
	xcov	互协方差函数估计
	cceps	复倒谱
	icceps	逆复倒谱
	rceps	实倒谱与线性相位重构
● 参数模型		
	arburg	Burg 法 AR 模型
	arcov	协方差法 AR 模型
	armcov	修正协方差法 AR 模型
	aryule	Yule-Walker 法 AR 模型
	invfreqs	模拟滤波器拟合频率响应
	invfreqz	离散滤波器拟合频率响应
	prony	Prony 法的离散滤波器拟合时间响应
	stmcb	Steiglitz-McBride 法求线性模型
● 线性预测		
	ac2rc	自相关序列到反射系数的变换
	ac2ploy	自相关序列变换为预测多项式
	is2rc	逆正弦参数变换为反射系数
	lar2rc	圆周率变换为反射系数
	levinson	Levinson-Durbin 递归算法
	Lpc	线性预测系数
	lsf2poly	线性谱频率变换为预测多项式
	Poly2ac	预测多项式变换为自相关序列
	Poly2lsf	预测多项式变换为线性谱频率
	Poly2rc	预测多项式变换为反射系数
	rc2ac	反射系数变换为自相关序列
	rc2is	反射系数变换为逆正弦参数

续表

● 线性预测		
	rc2lar	反射系数变换为圆周率
	rc2ploy	反射系数变换为预测多项式
	rlevison	逆 Levinson-Durbin 递归算法
	schurrc	Schur 算法
● 多采样率信号处理		
	decimate	以更低的采样频率重新采样数据
	Interp	以更高的采样频率重新采样数据
	interp1	一般的一维内插
	resample	以新的采样频率重新采样数据
	spline	三次样条内插
	upfirdn	FIR 的上下采样
● 波形产生		
	Chirp	产生调频波
	Diric	产生 Dirichlet 函数
	gauspuls	产生高斯射频脉冲
	gmonopuls	产生高斯单脉冲
	pulstran	产生脉冲串
	rectpuls	产生非周期的采样矩形脉冲
	sawtooth	产生锯齿波或三角波
	Sinc	产生 sinc 函数
	Square	产生方波
	Tripuls	产生非周期的采样三角形脉冲
	Vco	压控振荡器
● 特殊操作		
	Buffer	将信号矢量缓冲成数据矩阵
	cell2sos	将单元数组转换成二次矩阵
	cplxpair	将复数归成复共轭对
	Demod	通讯仿真中的解调
	Dpss	离散的扁球序列
	dpssclear	删除离散的扁球序列
	dpssdir	离散的扁球序列目录
	dpssload	装入离散的扁球序列
	dpsssave	保存离散的扁球序列
	eqtflength	补偿离散传递函数的长度
	modulate	通讯仿真中的调制

续表

● 特殊操作

	Seqperiod	寻找向量中重复序列的最小长度
	sos2cell	将二次矩阵转换成单元数组
	specgram	频谱分析
	Stem	画离散序列
	Strips	带形图
	Udecode	输入统一解码
	Uencode	输入统一编码

表 B. 10 小波工具箱

● 通用函数

	Biorfilt	双正交小波滤波器组
	Centfrq	小波中心频率
	dyaddown	二元取样
	Dyadup	二元插值
	Wavefun	小波函数与尺度函数
	Intwave	积分小波函数
	Orthfilt	正交小波滤波器组
	Qmf	镜像二次滤波器
	scal2frq	频率范围
	Wfilters	小波滤波器
	Wavem,gr	小波管理
	Wmaxlev	计算小波分解的最大尺度

● 小波函数

	biorwavf	双正交样条小波滤波器
	cgauwavf	复高斯小波
	cmorwavf	复 Morlet 小波
	coifwavf	Coiflets 小波滤波器
	Dbaux	Daubechies 小波滤波计算
	dbwavf	Daubechies 小波滤波器
	fbspwavf	复频率样条小波
	gauswavf	高斯小波
	mexihat	墨西哥草帽小波
	Meyer	Meyer 小波
	meyeraux	Meyer 小波辅助函数
	morlet	Morlet 小波
	rbiowavf	反双正交样条小波滤波器

续表

● 小波函数		
	Shanwavf	复 Shannon 小波
	symaux	Symlet 小波滤波器计算
	symwavf	Symlet 小波滤波器
● 一维小波变换		
	Cwt	一维连续小波变换
	appcoef	提取一维小波变换低频系数
	detcoef	提取一维小波变换高频系数
	Dwt	一维离散小波变换
	dwtmode	离散小波变换拓展模式
	Idwt	单尺度一维离散小波逆变换
	upcoef	一维系数的直接小波重构
	upwlev	单尺度一维小波分解的重构
	wavedec	多尺度一维小波分解
	waverec	多尺度一维小波重构
	wrcoef	对一维小波系数进行单支重构
● 二维小波变换		
	appcoef2	提取二维小波变换低频系数
	detcoef2	提取二维小波变换高频系数
	dwt2	二维离散小波变换
	dwtmode	离散小波变换拓展模式
	idwt2	单尺度二维离散小波逆变换
	upcoef2	二维系数的直接小波重构
	upwlev2	单尺度二维小波分解的重构
	wavedec2	多尺度二维小波分解
	waverec2	多尺度二维小波重构
	wrcoef2	对二维小波系数进行单支重构
● 小波包算法		
	bestlevt	计算完整最佳小波包数
	besttree	计算最佳数
	entrupd	更新小波包的熵值
	wentropy	计算小波包的熵
	wp2wtree	从小波包树中提取小波树
	Wpcoef	计算小波包系数
	wpcutree	剪切小波包分解树
	Wpdec	一维小波包分解

续表

● 小波包算法		
	Wpdec2	二维小波包分解
	Wpfun	小波包函数
	wpjoin	重新组合小波包
	wprcoef	小波包分解系数的重构
	Wprec	一维小波包分解的重构
	Wprec2	二维小波包分解的重构
	wpsplt	分割小波包
● 信号与图像的去噪与压缩		
	Ddencmp	获取默认值阈值、熵标准
	Thselect	信号去噪的阈值选择
	Wbmpen	一维与二维去噪的门限
	Wdcbm	Birge-Massart 规则一维去噪的门限
	Wdcbm2	Birge-Massart 规则二维去噪的门限
	Wden	用小波进行一维信号的自动去噪
	Wdencmp	用小波进行信号的去噪或压缩
	Wnoise	产生含噪声的测试函数数据
	Wnoisest	估计一维小波系数的偏差
	Wpbmpen	小波包去噪的门限
	Wpdencmp	用小波包进行信号的去噪或压缩
	Wpthcoef	进行小波包分解系数的阈值处理
● 信号与图像的消噪与压缩		
	wthcoef	一维信号的小波系数阈值处理
	wthcoef2	二维信号的小波系数阈值处理
	wthresh	进行软阈值或硬阈值处理
	wthrmngr	门限设置管理
● 树操作应用函数		
	Allnodes	计算树结点
	depo2ind	结点深度-位置形式转化成索引形式
	drawtree	画出小波包分解树
	Dtree	类 DTREE 的构造器
	Get	获取树对象的内容
	ind2depo	结点索引形式转化成深度-位置形式
	Isnode	判断结点是否存在
	Istnode	判断结点是否是终点并返回排列值
	Leaves	确定终结点

续表

● 树操作应用函数		
	Nodcasc	计算上溯结点
	nodedesc	计算下溯结点
	nodejoin	重组结点
	Nodepar	寻找父结点
	nodesplt	分割结点
	noleaves	确定非终结点
	Ntnode	求终结点的个数
	Ntree	类 NTREE 的构造器
	Plot	画出树对象
	Read	读取树对象的值
	readtree	从图形中读取小波包分解树
	Set	设置树对象的内容
	Tnodes	确定终结点
● 树操作应用函数		
	Treedpth	求树的深度
	Treeord	求树结构的叉数
	Wptree	类 WPTREE 的构造器
	wpviewcf	画出小波包颜色系数
	Write	在树对象域写值
	Wtbo	类 WTBO 的构造器
	wtreemgr	管理树结构
● 其他		
	wcodemat	扩展伪颜色矩阵
	Wextend	扩展向量或矩阵
	Wkeep	提取向量或矩阵中的一部分
	Wrev	向量逆序
	instdfft	非标准一维快速傅立叶逆变换
	Nstdfft	非标准一维快速傅立叶变换
	Wvarchg	寻找变量改变点